Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων Πληροφορικης

Ανίχνευση Ανωμαλιών σε Δίκτυα Υπολογιστών με Χρήση Τομογραφίας Δικτύου

Διπλωματικη Εργασια

του

Δημητρίου Γ. Γιαννή

Επιβλέπων: Συμεών Παπαβασιλείου
Καθηγητής ΕΜΠ

Αθήνα, Οκτώβριος 2021

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων Πληροφορικης

Ανίχνευση Ανωμαλιών σε Δίκτυα Υπολογιστών με Χρήση Τομογραφίας Δικτύου

Διπλωματικη Εργασια

του

Δημητρίου Γ. Γιαννή

Επιβλέπων: Συμεών Παπαβασιλείου
Καθηγητής ΕΜΠ

Εγκρίθηκε από την κάτωθι τριμελή επιτροπή την 3η Νοεμβρίου 2021.

Συμεών Παπαβασιλείου	Θεοδώρα Βαρβαρίγου	Βασίλειος Καρυώτης
Καθηγητής ΕΜΠ	Καθηγήτρια ΕΜΠ	Αν. Καθηγητής Ιονίου Παν/μίου

Δημήτριος Γιαννής
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών ΕΜΠ

Περίληψη

Στη σημερινή εποχή το Διαδίκτυο αποτελεί όλο και περισσότερο αναπόσπαστο κομμάτι κάθε ατομικής και κοινωνικής δραστηριότητας. Ωστόσο, στο πλαίσιο της νέας αυτής πραγματικότητας, τόσο το μέγεθος όσο και η πολυπλοκότητα του Διαδικτύου αυξάνονται, δημιουργώντας, έτσι, την ανάγκη για αποδοτικότερη επιτήρηση (monitoring) των επιμέρους (υπο)δικτύων που το συνθέτουν. Λαμβάνοντας υπόψη τα παραπάνω, η μέθοδος της Τομογραφίας Δικτύου αποτελεί μία πρωτοπόρα μέθοδο επιτήρησης, η οποία επιτυγχάνει την ακριβή εξαγωγή των ζητούμενων πληροφοριών του δικτύου (τοπολογία και εσωτερικές παράμετροι) με έμμεσο τρόπο, δηλαδή μέσω μετρήσεων στα άκρα του. Στην παρούσα διπλωματική εργασία, αρχικά, μελετήθηκαν διεξοδικά οι βασικές τεχνικές Τομογραφίας Δικτύου. Στη συνέχεια, και με βάση την υπάρχουσα βιβλιογραφία, επιλέχθηκε και μελετήθηκε ένας συγκεκριμένος τομογραφικός αλγόριθμος, ο οποίος επιτρέπει την ανακατασκευή ολόκληρης της τοπολογίας, καθώς και τον υπολογισμό των βασικών χαρακτηριστικών κάθε ακμής της. Έπειτα, πραγματοποιήθηκε μία θεωρητική επισκόπηση των μεθόδων στατιστικής επεξεργασίας δεδομένων offline change point detection. Όσον αφορά το πειραματικό μέρος της εργασίας, χρησιμοποιώντας το εργαλείο πειραματικών δοκιμών jFed, υλοποιήθηκε μία εφαρμογή δικτυακής επιτήρησης, η οποία αξιοποιεί τον προαναφερθέντα αλγόριθμο για την εξαγωγή αποτελεσμάτων, δημιουργεί χρονοσειρές (time series) για την απώλεια πακέτων σε κάθε ακμή της τοπολογίας και εν τέλει πραγματοποιεί στατιστική επεξεργασία των χρονοσειρών, μέσω μεθόδων offline change point detection. Το τελικό αποτέλεσμα είναι ο εντοπισμός χρονικών στιγμών για τις οποίες παρατηρείται μεγάλη απόκλιση στην απώλεια πακέτων, υποδηλώνοντας με αυτόν τον τρόπο την ύπαρξη ενδεχόμενων ανωμαλιών στο δίκτυο (network anomalies). Η συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιηθεί σε πραγματικές δικτυακές τοπολογίες, δίνοντας λύση σε μία σειρά από προβλήματα, όπως η εξασφάλιση της σωστής λειτουργίας του εκάστοτε δικτύου, αλλά και η ανίχνευση ενδεχόμενων επιθέσεων σε αυτό.

Λέξεις-κλειδιά: δικτυακή επιτήρηση, Τομογραφία Δικτύου, jFed, offline change point detection, ανωμαλίες δικτύου

Abstract

In today's era, the Internet is becoming more and more an integral part of almost every individual and social activity. In the framework of this new reality, both the size and the complexity of Internet increase, creating therefore the need for more efficient monitoring of the subnetworks that compose it. By taking into account the above facts, the method of Network Tomography constitutes a pioneering and accurate monitoring method, which infers the desired network information (topology and internal characteristics) indirectly, that is, through end-to-end measurements. In the framework of this diploma thesis, the basic techniques of Network Tomography were studied thoroughly. Thereafter, based on the existing literature, a specific tomographic algorithm was selected and studied, as it enables the inference of the whole topology and the calculation of the basic characteristics for each of its internal edges. Next, the offline change point detection methods for statistical analysis were presented. As far as the experimental part of the thesis is concerned, the jFed experimental testing tool was used in order to create a network monitoring application, which in turn uses the above algorithm to produce results, creates time series for the packet loss on each edge of the topology and finally processes the time series through offline change point detection methods. The final result is the detection of moments in which the packet loss presents a substantial deviation, therefore indicating the existence of possible network anomalies. This specific application can be used in real world network topologies, thus providing solution to a series of problems, such as the maintenance of the network's normal function or the detection of attacks against it.

Keywords: network monitoring, Network Tomography, jFed, offline change point detection, network anomalies

Ευχαριστίες

Κατ' αρχάς, θα ήθελα να ευχαριστήσω θερμά τον καθηγητή ΕΜΠ Συμεών Παπαβασιλείου για την καθοδήγηση και την εμπιστοσύνη που μου έδειξε καθ' όλη τη διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας, καθώς και για τις πολύτιμες συμβουλές του σε ακαδημαϊκό και επαγγελματικό επίπεδο.

Θα ήθελα, επίσης, να ευχαριστήσω ιδιαίτερα τον Αναπληρωτή Καθηγητή του Τμήματος Πληροφορικής του Ιονίου Παν/μίου Βασίλη Καρυώτη καθώς και τον διδακτορικό φοιτητή ΕΜΠ Γρηγόρη Κακκάβα για τον χρόνο που μου αφιέρωσαν, τη διαρκή τους υποστήριξη και το ευχάριστο κλίμα που δημιούργησαν, οδηγώντας κατά αυτόν τον τρόπο σε μία εξαιρετική συνεργασία.

Στη συνέχεια, θα ήθελα να ευχαριστήσω τους γονείς μου και τον αδελφό μου, οι οποίοι ήταν δίπλα μου σε κάθε βήμα μου έως τώρα. Χωρίς εκείνους δεν θα βρισκόμουν στο σημείο όπου βρίσκομαι σήμερα.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου και τα αγαπημένα μου πρόσωπα, με τα οποία μοιράστηκα ορισμένες από τις ομορφότερες στιγμές της φοιτητικής μου ζωής και τα οποία βρίσκονταν πάντα εκεί για εμένα.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Πρόλογος

Η ολοένα και μεγαλύτερη ενσωμάτωση του Διαδικτύου σε ολόκληρο το φάσμα της ανθρώπινης δραστηριότητας έχει ως αναπόφευκτο αποτέλεσμα την ταχύτατη εξέλιξή του τόσο ως προς την έκτασή του όσο και ως προς τη συνθετότητά του. Συγκεκριμένα, η ίδια η **δομή του Διαδικτύου** αποκτά έναν διαρκώς **αποκεντρωμένο και ετερογενή χαρακτήρα**, στον οποίον κυριαρχεί η ύπαρξη πολλαπλών, ενίοτε μη συνεργαζόμενων υποδικτύων, τα οποία υπάγονται σε διαφορετική διαχείριση [1]. Σε αυτό το νέο διαδικτυακό περιβάλλον, ζητήματα όπως η αποδοτικότητα των παρεχόμενων υπηρεσιών, καθώς και η ασφάλεια των χρησιμοποιούμενων δικτυακών υποδομών φαντάζουν πιο επιτακτικά από ποτέ. Η εξασφάλιση όλων των παραπάνω δημιουργεί αναπόφευκτα την ανάγκη για εξέλιξη και βελτίωση των υπαρχόντων μεθόδων και εργαλείων δικτυακής επιτήρησης (network monitoring). Με τον όρο δικτυακή επιτήρηση ορίζουμε τη χρήση υπολογιστικών συστημάτων, τα οποία παρέχουν στους διαχειριστές του δικτύου τις απαραίτητες πληροφοριές, ώστε να αποφανθούν για τη σωστή λειτουργία του [2].

Δυστυχώς, οι πιο διαδεδομένες μέθοδοι επιτήρησης που χρησιμοποιούνται σήμερα βασίζονται σε μία σειρά από προϋποθέσεις, όπως η συνεργασία μεταξύ όλων των εμπλεκόμενων κόμβων του δικτύου ή η προσβασιμότητα σε αυτούς. Προϋποθέσεις, οι οποίες με βάση την εξελικτική πορεία του Διαδικτύου δεν είναι πλέον καθόλου δεδομένες [1]. Λαμβάνοντας υπόψη τις παραπάνω δυσκολίες, η **Τομογραφία Δικτύου** αποτελεί μία σχετικά πρόσφατη και πρωτοπόρα μέθοδο δικτυακής επιτήρησης, η οποία επιχειρεί να εξάγει όλες τις ζητούμενες πληροφορίες (τοπολογία και εσωτερικές παράμετροι) για το δίκτυο, **χωρίς καμία προηγούμενη απαίτηση από τα επιμέρους στοιχεία του**, πέρα από την απλή προώθηση πακέτων [3], [4].

1.2 Αντικείμενο της Διπλωματικής

Με αφορμή τα όσα αναφέρθηκαν στην προηγούμενη ενότητα, η παρούσα Διπλωματική επιχειρεί να αξιοποιήσει τις δυνατότητες που προσφέρει η Τομογραφία Δικτύου, δίνοντας, έτσι, μία απάντηση στην έλλειψη σύγχρονων και αποτελεσματικών μεθόδων δικτυακής επιτήρησης. Για τον σκοπό αυτό, το αντικείμενό της συνίσταται, κατ' αρχάς, στη θεωρητική επισκόπηση των διαφόρων τεχνικών **Τομογραφίας Δικτύου**, καθώς και των μεθόδων στατιστικής ανάλυσης **offline change point detection**. Κατά δεύτερον, στη **δημιουργία μίας πραγματικής πειραματικής τοπολογίας δικτύου**, δεσμεύοντας τους κατάλληλους υπολογιστικούς πόρους μέσω του συστήματος δοκιμών που υποστηρίζει το jFed. Τέλος, στην **υλοποίηση μιας εφαρμογής**, η οποία δοκιμάστηκε πάνω στην εν λόγω τοπολογία και η οποία βασίζεται στις δύο παραπάνω τεχνολογίες, με σκοπό την **ανίχνευση ανωμαλιών δικτύου**. Συγκεκριμένα, η Τομογραφία Δικτύου χρησιμοποιείται από την εφαρμογή για την επιτήρηση (monitoring) του δικτύου, παράγοντας αποτελέσματα σχετικά με την απώλεια πακέτων στις ζεύξεις του. Η πληροφορία αυτή αξιοποείται στη συνέχεια για τη δημιουργία χρονοσειρών (time series), οι οποίες με τη σειρά τους αποτελούν τα δεδομένα πάνω στα οποία θα εφαρμοστούν οι μέθοδοι offline change point detection. Αποτέλεσμα είναι η εκτίμηση των εκάστοτε σημείων αλλαγής (change points) στην απώλεια των ζεύξεων, τα οποία υποδηλώνουν πιθανές ανωμαλίες δικτύου (network anomalies).

Με τον όρο **ανωμαλίες δικτύου** ορίζονται χρονικές στιγμές, οι οποίες χαρακτηρίζονται από απότομες και σχετιζόμενες μεταξύ τους **αλλαγές στα μετρήσιμα μεγέθη που προσδιορίζουν την επίδοση του δικτύου**. Χαρακτηριστικά παραδείγματα τέτοιων μεγεθών είναι η χρονική καθυστέρηση και η απώλεια των πακέτων στις ζεύξεις της τοπολογίας. Στο διεξαγόμενο πείραμα, μάλιστα, οι αλλαγές που συνιστούν τα σημεία αλλαγής προκύπτουν μέσα από μεταβολές της κατανομής, βάσει της οποίας ανατίθενται οι τιμές απώλειας σε κάθε ζεύξη. Το τελικό αποτέλεσμα είναι στην ουσία καταστάσεις όπου οι λειτουργίες του δικτύου απέχουν από το φυσιολογικό επίπεδο [5]. Η ανίχνευση δε των εν λόγω καταστάσεων αποτελεί βασική προϋπόθεση τόσο για την εξασφάλιση της ομαλής λειτουργίας των δικτυακών υποδομών, όσο και για την αναγνώριση πιθανών επιθέσεων. Κι αυτό, διότι οι δικτυακές επιθέσεις σε πολλές περιπτώσεις έχουν άμεση επίπτωση στην επίδοση των δικτύων που πλήττουν.

1.3 Δομή της Διπλωματικής

Στο κεφάλαιο 2, αρχικά, παρουσιάζεται και αναλύεται η μέθοδος της Τομογραφίας Δικτύου μαζί με τις τρεις βασικές κατηγορίες της. Στη συνέχεια, μελετάται συνοπτικά ο αλγόριθμος Τομογραφίας Δικτύου, ο οποίος επιστρατεύτηκε για την υλοποίηση της εφαρμογής. Στο κεφάλαιο 3 πραγματοποιείται μία θεωρητική επισκόπηση των μεθόδων στατιστικής ανάλυσης offline change point detection. Στο κεφάλαιο 4 περιγράφεται το εργαλείο πειραματικών δοκιμών jFed, καθώς και τα υπόλοιπα εργαλεία που χρησιμοποιήθηκαν στο πειραματικό μέρος. Στο κεφάλαιο 5 παρουσιάζεται η λογική της εφαρμογής, μέσω του σχετικού αλγορίθμου, ενώ στο κεφάλαιο 6 παρατίθενται τα αποτελέσματα, τα οποία παράχθηκαν ανάλογα με τις εκάστοτε παραμετροποιήσεις, καθώς επίσης και η συγκριτική τους μελέτη. Στο κεφάλαιο 7 καταγράφονται τα συμπεράσματα που προέκυψαν από την παρούσα διπλωματική εργασία, ενώ αναφέρονται κι ορισμένα ανοιχτά προβλήματα που εμπίπτουν στο πεδίο της συγκεκριμένης εφαρμογής. Ακολουθεί η λίστα αναφορών για τη σχετική βιβλιογραφία και στη συνέχεια το Παράρτημα, όπου παρατίθενται ο κώδικας που εκτελέστηκε από το κάθε στοιχείο της πειραματικής τοπολογίας.

Κεφάλαιο 2

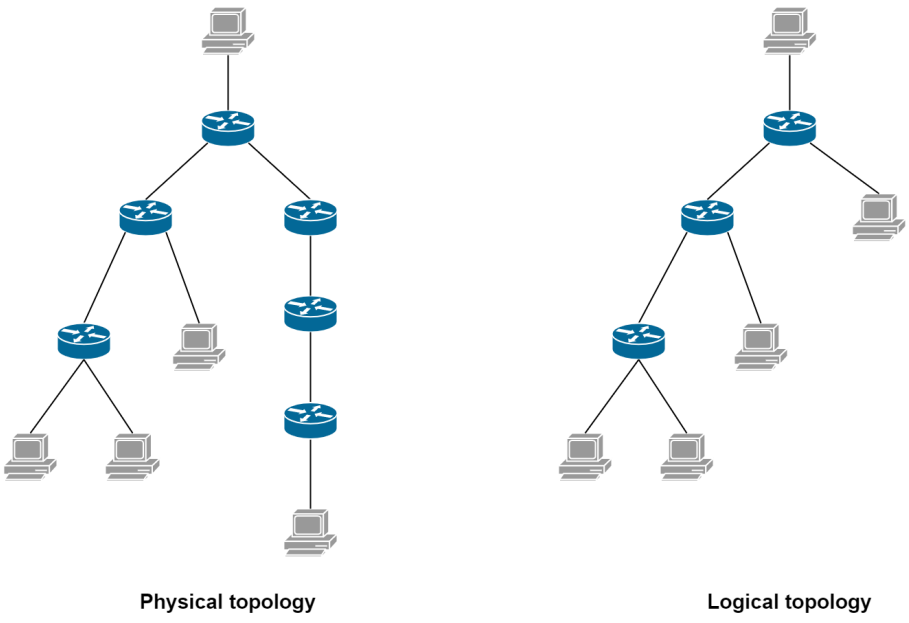
Τομογραφία Δικτύου

2.1 Γενικά

Όπως αναφέρθηκε και προηγουμένως, η μέθοδος Τομογραφίας Δικτύου (Network Tomography), ως μέθοδος επιτήρησης, επιχειρεί να εξάγει την τοπολογία του δικτύου, καθώς και εσωτερικές παραμέτρους που αφορούν την επίδοσή του (για παράδειγμα απώλεια πακέτων ή καθυστέρηση ανά ζεύξη), αξιοποποιώντας όσο το δυνατό λιγότερες προϋποθέσεις και γνώσεις για αυτό. Προκειμένου να επιτύχει κάτι τέτοιο, εκμεταλλεύεται τη δικτυακή κίνηση μεταξύ κόμβων που βρίσκονται στα άκρα της τοπολογίας, γνωστοί και ως επιτηρητές (monitors). Στην ουσία, καταγράφοντας τα πακέτα που στέλνονται μεταξύ των επιτηρητών από το ένα άκρο του δικτύου στο άλλο, προκύπτουν μετρήσεις, γνωστές ως ακραίες μετρήσεις **(end-to-end measurements)** και, στη συνέχεια, με βάση αυτές, εξάγονται **έμμεσα** η **τοπολογία** και οι **εσωτερικές παράμετροι** [6]. Αξίζει, μάλιστα, να σημειωθεί ότι, αναλόγως τις εκάστοτε απαιτήσεις, είναι δυνατή και η αντίστροφη λειτουργία, δηλαδή η εξαγωγή πληροφορίας για τα άκρα της τοπολογίας, βάσει μετρήσεων στο εσωτερικό της [1], [4].

2.2 Ορισμός του προβλήματος

Κάθε υποδίκτυο που υπάγεται στο Διαδίκτυο μπορεί να μοντελοποιηθεί ως ένας **γράφος**, του οποίου οι κόμβοι αναπαριστούν τα επιμέρους στοιχεία του, όπως τελικοί χρήστες (clients), δρομολογητές (routers), εξυπηρετητές (servers) και μεταγωγείς (switches). Κάθε ακμή (link) του γράφου αναπαριστά μια λογική σύνδεση (ζεύξη) μεταξύ δύο κόμβων, η οποία με τη σειρά της μπορεί να περιλαμβάνει μία ή περισσότερες φυσικές συνδέσεις. Χαρακτηριστικό πάραδειγμα είναι η περίπτωση διαδοχικής σύνδεσης κόμβων χωρίς διακλαδώσεις, η οποία απεικονίζεται στο σχήμα 2.1 [7]. Στο συγκεκριμένο σχήμα, με τον όρο ¨φυσική τοπολογία¨ (**physical topology**) ορίζεται η απεικόνιση του δικτύου, βάσει των φυσικών συνδέσεών του, ενώ με τον όρο ¨λογική τοπολογία¨ (**logical topology**), η απεικόνιση του, βάσει των αντίστοιχων λογικών συνδέσεων. Η διαφορά των δύο τοπολογιών έγκειται στο γεγονός ότι, στην περίπτωση της λογικής τοπολογίας, οι εσωτερικοί κόμβοι που αναπαρίστανται είναι μόνο εκείνοι, οι οποίοι αποτελούν ¨σημεία διακλάδωσης¨ [8].



Σχήμα 2.1: Σύγκριση φυσικής και λογικής τοπολογίας

Αξίζει να σημειωθεί, επίσης, ότι κάθε ακμή μπορεί να είναι μονής ή διπλής κατεύθυνσης, ανάλογα με τον βαθμό αφαιρετικότητας και τα χαρακτηριστικά της τοπολογίας που μοντελοποιείται, ενώ το σύνολο ενός ή περισσότερων διαδοχικών ακμών μεταξύ δύο κόμβων αποτελεί ένα μονοπάτι (path) του γράφου [1].

Με βάση, λοιπόν, τους παραπάνω ορισμούς, η έμμεση εξαγωγή της τοπολογίας και των εσωτερικών παραμέτρων αποτελεί στην πραγματικότητα ένα ¨αντίστροφο στατιστικό πρόβλημα¨ (**statistical inverse problem**), το οποίο μπορεί να περιγραφεί μέσω της γραμμικής σχέσης:

$$Y_t = AX_t + e \qquad \text{(I)}$$

Στην παραπάνω σχέση, A είναι ο πίνακας δρομολόγησης, διαστάσεων $I \times J$ (I: ο αριθμός των διαφορετικών μονοπατιών και J: ο αριθμός των διαφορετικών ζεύξεων). Κάθε στοιχείο (i,j) του A λαμβάνει την τιμή 1 εάν η ζεύξη μεταξύ των κόμβων i,j περιλαμβάνεται σε κάποιο από τα I μονοπάτια, ενώ σε διαφορετική περίπτωση λαμβάνει την τιμή 0. Αντίστοιχα, Y_t είναι ένας I-διάστατος πίνακας, ο οποίος περιέχει τις μετρήσεις για μια δεδομένη χρονική στιγμή t, ενώ X_t είναι ένας J-διάστατος πίνακας, ο οποίος περιέχει τις εσωτερικές παραμέτρους (όπως η μέση καθυστέρηση των πακέτων). Τέλος, η παράμετρος e είναι η παράμετρος του θορύβου [4]. Η λύση, επομένως, του προβλήματος έγκειται στην επίλυση της εν λόγω γραμμικής σχέσης ως προς τον άγνωστο (για τον διαχειριστή) πίνακα X_t, δεδομένου του γνωστού πίνακα Y_t.

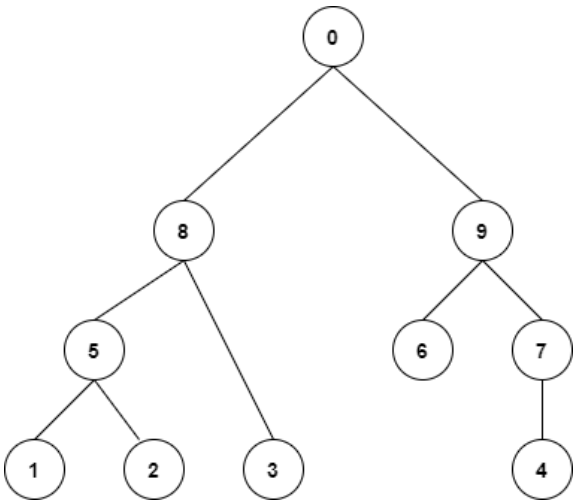
2.3 Βασικές κατηγορίες

Οι μέθοδοι Τομογραφίας Δικτύου που συναντά κανείς στη διαθέσιμη βιβλιογραφία χωρίζονται σε **τρείς βασικές κατηγορίες**. Με βάση τις παραπομπές [1], [4], οι κατηγορίες αυτές είναι οι ακόλουθες:

- Μέθοδοι που εκτιμούν τις παραμέτρους του δικτύου σε επίπεδο ζεύξης, βάσει ακραίων μετρήσεων σε επίπεδο μονοπατιού (**link-level parameter estimation**). Σε αυτήν την κατηγορία ο πίνακας Y_t περιέχει τις γνωστές ακραίες μετρήσεις (όπως ο αριθμός των πακέτων που χάνονται από το ένα άκρο του δικτύου στο άλλο) και ο πίνακας X_t τις άγνωστες παραμέτρους επίδοσης κάθε ζεύξης (όπως το ποσοστό απωλειών της). Οι τελευταίες μάλιστα τείνουν να είναι αθροιστικές με την έννοια ότι η παράμετρος επίδοσης ενός μονοπατιού της τοπολογίας προκύπτει από το άθροισμα των παραμέτρων επίδοσης κάθε επιμέρους ζεύξης του μονοπατιού.

  Αντιπροσωπευτικό παράδειγμα της εν λόγω κατηγορίας είναι η μέθοδος που παρουσιάζεται στην παραπομπή [9]. Η μέθοδος αυτή αξιοποιεί **την εκπομπή**

τύπου unicast από έναν κόμβο-πηγή (source) προς έναν κόμβο-προορισμό[1] της τοπολογίας, προκειμένου να δημιουργήσει δικτυακή κίνηση από άκρο σε άκρο του δικτύου και κατ' επέκταση ακραίες μετρήσεις. Με βάση, λοιπόν, το ποσοστό απωλειών που προκύπτει από τις ακραίες μετρήσεις για κάθε μονοπάτι μεταξύ πηγής-προορισμού, επιχειρείται η εξαγωγή του ποσοστού απωλειών για τις επιμέρους ζεύξεις του δικτύου. Αυτό επιτυγχάνεται στέλνοντας σε σύντομο χρονικό διάστημα δύο διαδοχικά πακέτα (**back-to-back packets**), τα οποία έχουν διαφορετικό προορισμό, αλλά τα μονοπάτια που ακολουθούν προς αυτόν έχουν ορισμένες κοινές ζεύξεις. Έτσι, δεδομένου ότι δύο τέτοια πακέτα ακολουθούν κοινή διαδρομή μέχρι τον κόμβο u και στη συνέχεια διαχωρίζονται, ώστε το καθένα να φτάσει σε ένα από τα δύο "παιδιά" του i, j, εάν το ένα από τα πακέτα φτάσει στον i, μπορεί να θεωρηθεί ότι και τα δύο φτάσανε στον u. Επομένως, η μη άφιξη του άλλου πακέτου στον j υποδηλώνει απώλεια στη ζεύξη $u - j$. Στέλνοντας, λοιπόν, διαδοχικά πακέτα σε διαφορετικούς προορισμούς i, j και εφαρμόζοντας τον παραπάνω συλλογισμό είναι δυνατό να απομονωθεί το ποσοστό απωλειών για κάθε επιμέρους ζεύξη της τοπολογίας. Ο παραπάνω συλλογισμός μπορεί να εφαρμοστεί, για παράδειγμα, στην ακόλουθη τοπολογία:



Σχήμα 2.2: Link-level parameter estimation με χρήση back-to-back packets

Έστω, ότι δύο back-to-back packets στέλνονται από την πηγή (κόμβος 0) με το ένα να έχει προορισμό τον κόμβο 1 και το άλλο τον κόμβο 2. Στην περίπτωση αυτή, τα δύο πακέτα ακολουθούν κοινό μονοπάτι από τον κόμβο 0 έως τον κόμβο 5, στον οποίο και διαχωρίζονται. Κατά συνέπεια εάν το πρώτο πακέτο φτάσει στον κόμβο 1 (άρα και στον προηγούμενο κόμβο 5), μπορεί να θεωρηθεί ότι και το δεύτερο πακέτο έφτασε στον κόμβο 5, καθώς τα δύο πακέτα

[1] Ο κόμβος-πηγή και οι κόμβοι-προορισμοί βρίσκονται στα άκρα της τοπολογίας με τον κόμβο-πηγή να τοποθετείται στην κορυφή του γράφου και τους κόμβους-προορισμούς στα "φύλλα" του.

στέλνονται με πολύ μικρή χρονική διαφορά. Εφόσον, λοιπόν, και τα δύο πακέτα φτάσανε στον κόμβο 5, η μη άφιξη του δεύτερου πακέτου στον κόμβο 2 συνεπάγεται απώλεια στη ζεύξη $5 - 2$.

Μία διαφορετική υλοποίηση, η οποία επιστρατεύει **την εκπομπή τύπου multicast** έναντι της εκπομπής unicast, περιγράφεται αναλυτικά στην παραπομπή [10]. Συγκεκριμένα, η απώλεια ενός πακέτου σε μια ζεύξη της τοπολογίας αντιμετωπίζεται ως μία διαδικασία, τέτοια ώστε το σύνολο των αντίστοιχων διαδικασιών όλων των ζεύξεων να είναι ένα σύνολο ανεξάρτητων μεταξύ τους στοιχείων που ακολουθούν κατανομή Bernoulli. Για κάθε κόμβο k ορίζεται η πιθανότητα α_k ενός δεδομένου πακέτου να μην χαθεί στη ζεύξη που καταλήγει σε αυτόν τον κόμβο. Η δε **διαδρομή ενός πακέτου από την πηγή προς έναν κόμβο** μπορεί να θεωρηθεί ως μια **στοχαστική διαδικασία** $X = (X_k)$, όπου $X_k = 1$ αν το πακέτο έφτασε στον κόμβο k και $X_k = 0$ αν δεν έφτασε. Δεδομένου ότι μόνο ακραίες μετρήσεις είναι δυνατές, το αποτέλεσμα της αποστολής ενός πακέτου αναπαρίσταται από ένα σύνολο τιμών $X = (X_r)$, όπου r ανήκει στο σύνολο των κόμβων-προορισμών. Επομένως, για ένα δεδομένο set πιθανοτήτων $\alpha = (\alpha_k)$, η πιθανότητα να προκύψει ως αποτέλεσμα ένα συγκεκριμένο σύνολο τιμών $X = (X_r)$ (το οποίο συμβολίζεται ως x) είναι $p(x|\alpha)$. Αντίστοιχα, στην περίπτωση αποστολής n πακέτων από την πηγή, η πιθανότητα να προκύψουν τα n ανεξάρτητα αποτελέσματα $x^1, ..., x^n$ είναι $p(x^1, ..., x^n|\alpha)$. Το πρόβλημα, λοιπόν, ανάγεται στην εκτίμηση του μεγέθους α ($\hat{\alpha}$), έτσι ώστε να μεγιστοποείται η πιθανότητα $p(x^1, ..., x^n|\alpha)$ και κατ' επέκταση η συνάρτηση $L(\alpha) = logp(x^1, ..., x^n|\alpha)$.

- Μέθοδοι που εκτιμούν τη δικτυακή κίνηση μεταξύ αποστολέα-παραλήπτη σε επίπεδο μονοπατιού, βάσει μετρήσεων σε επίπεδο ζεύξης (**path-level traffic intensity estimation**). Σε αυτήν την κατηγορία στόχος είναι η εξαγωγή της κίνησης/κυκλοφορίας από το ένα άκρο του δικτύου στο άλλο και για κάθε συνδυασμό αποστολέα-παραλήπτη, καταγράφοντας την κίνηση σε μεμονωμένες ζεύξεις. Συγκεκριμένα, για κάθε μεμονωμένη ζεύξη, καταγράφονται τα πακέτα που διέρχονται από τους αντίστοιχους κόμβους κι έτσι προκύπτει ο γνωστός πίνακας Y_t. Όταν εκτιμηθεί η δικτυακή κίνηση για κάθε συνδυασμό αποστολέα-δέκτη, σχηματίζεται ο άγνωστος αρχικά πίνακας κίνησης (traffic matrix) X_t.

Μία χαρακτηριστική μέθοδος που εμπίπτει σε αυτήν την κατηγορία είναι η μέθοδος της παραπομπής [11]. Στην προκειμένη περίπτωση, χρησιμοποιώντας τις μετρήσεις που είναι διαθέσιμες για κάθε ζεύξη του δικτύου από

το πρωτόκόλλο SNMP[2], επιλύεται ένα ¨μοντέλο βαρύτητας¨ (**gravity model**). Πρόκειται για ένα μοντέλο, το οποίο συσχετίζει τη δικτυακή κίνηση από μία ζεύξη l_i προς μία άλλη ζεύξη l_j ($T(l_i, l_j)$) με τη συνολική κίνηση που εισέρχεται στην πρώτη ζεύξη και τη συνολική κίνηση που εξέρχεται από τη δεύτερη. Από την παραπάνω επίλυση προκύπτει μία **αρχική λύση t_g**, η οποία με τη σειρά της οδηγεί στην **ακριβέστερη λύση t**. Η τελευταία αποτελεί μία βελτιωμένη ¨εκδοχή¨ της t_g, τέτοια, ώστε αφενός να έχει τη μικρότερη δυνατή απόκλιση από αυτήν κι αφετέρου να πληροί την εξίσωση της τομογραφίας δικτύου. Στην ουσία πρόκειται για την επίλυση του ακόλουθου προβλήματος:

$$min||t - t_g|| \quad \text{έτσι ώστε} \quad ||At - x|| \quad \text{να ελαχιστοποιείται}$$

όπου $x = At$: η απλοποιημένη μορφή της εξίσωσης (I) για $x \equiv Y_t$ και $t \equiv X_t$.

Ένα ακόμη παράδειγμα αυτής της κατηγορίας μεθόδων με ιδιαίτερη αποτελεσματικότητα σε μεγάλα δίκτυα είναι εκείνο της παραπομπής [12]. Η βασική ιδέα της συγκεκριμένης μεθόδου έγκειται στα ακόλουθα βήματα. Κατ' αρχάς, όλα τα **ζεύγη κόμβων της μορφής (πηγή, προορισμός)** ομαδοποιούνται σε **σύνολα (sets)** ($S_1.S_2, ..., S_M$), τα οποία είναι **ξένα μεταξύ τους**. Δηλαδή, όλα τα ζεύγη που μοιράζονται κοινή πηγή ή κοινό προορισμό τοποθετούνται στο ίδιο set. Έπειτα, για κάθε ζεύγος ενός set S επιλέγεται ένα σύνολο από μετρήσεις επιπέδου ζεύξης και με βάση αυτές κατασκευάζεται μία **ελαχιστοποιημένη μορφή** της σχέσης $Y_t = AX_t$ (ενότητα 2.2), από την οποία προκύπτουν οι εκτιμήσεις για τις εσωτερικές παραμέτρους (X_t). Στην πιο απλή (αλλά λιγότερο βέλτιση) μορφή της, η μέθοδος αυτή εξετάζει κάθε ζεύγος πηγής-προορισμού (o, d) ξεχωριστά (όχι σε sets) και αξιοποιεί ως μετρήσεις τα πακέτα που ξεκινούν από τον κόμβο o ($y[o]$) και τα πακέτα που προορίζονται για τον κόμβο d ($y[d]$). Με βάση τα παραπάνω, ορίζονται ως $y[od]$: τα πακέτα που ξεκινούν από τον κόμβο o και προορίζονται για τον d, ως $y[o\overline{d}]$: τα πακέτα που ξεκινούν από τον κόμβο o αλλά προορίζονται για οποιονδήποτε άλλον εκτός του d και ως $y[\overline{o}d]$: τα πακέτα που προορίζονται για τον κόμβο d, αλλά ξεκινούν από οποιονδήποτε άλλον εκτός του o. Επομένως, προκύπτει η εξής απλοποιημένη μορφή του πίνακα $Y_t = AX_t$:

[2]Πρωτόκολλο διαχείρισης δικτύου (**Network Management Protocol**), το οποίο βασίζεται στο μοντέλο client-server. Στον ρόλο του server είναι ο SNMP manager και στον ρόλο του client κάθε host του δικτύου. Κάθε host έχει έναν SNMP agent που ελέγχεται από τον SNMP manager, ο οποίος με τη σειρά του διατηρεί μία βάση μεταβλητών (MIB). Οι μεταβλητές αυτές χαρακτηρίζουν τις λειτουργίες των agents και κάθε agent έχει ένα αντίστοιχο set τιμών γι αυτές τις μεταβλητές. [5]

$$\begin{pmatrix} y[o] \\ y[d] \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} y[od] \\ y[o\overline{d}] \\ y[\overline{o}d] \end{pmatrix}$$

Από την παραπάνω σχέση είναι δυνατό να υπολογιστούν οι εκτιμήσεις των μεγεθών $y[od], y[o\overline{d}], y[\overline{o}d]$ και συγκεκριμένα της $y[od]$. Η εκτίμηση της $y[od]$ δίνεται από τη σχέση $\lambda[od] = E(y[od])$, όπου E( ): η εκτιμώμενη (μέση) τιμή. Επαναλαμβάνοντας την παραπάνω διαδικασία για κάθε ζεύγος πηγής-προορισμού, υπολογίζεται η ανάλογη εκτίμηση για κάθε ζεύξη της τοπολογίας.

- Μέθοδοι που εξάγουν ολόκληρη την τοπολογία του δικτύου (**topology inference**). Σε αυτήν την κατηγορία η τοπολογία του δικτύου και κατ' επέκταση ο πίνακας A που την εκφράζει είναι άγνωστη. Ο προσδιορισμός της τοπολογίας, λοιπόν, επιτυγχάνεται μέσω ακραίων μετρήσεων στο δίκτυο, με βάση τις οποίες προσδιορίζεται η ομοιότητα μεταξύ δύο κόμβων-προορισμών στα άκρα της τοπολογίας, δηλαδή το ποσοστό των κοινών ζεύξεων στα μονοπάτια που οδηγούν από τον αποστολέα σε καθέναν από τους δύο κόμβους. Στην ουσία, όσο περισσότερες είναι οι παραπάνω κοινές ζεύξεις, τόσο μεγαλύτερη θα είναι η ομοιότητα των μετρήσεων που λαμβάνουμε για αυτούς. Οπότε, με βάση την ομοιότητα των μετρήσεων, μπορούμε να συμπεράνουμε την ομοιότητα (κι άρα την εγγύτητα) μεταξύ των δύο κόμβων στη λογική τοπολογία. Ακολουθώντας αυτό το σκεπτικό για ζευγάρια κόμβων-προορισμών και αξιοποιώντας στατιστικές μεθόδους (όπως hierarchical clustering, maximum likelihood, Bayesian inference) είναι δυνατός ο προσδιορισμός ολόκληρης της λογικής τοπολογίας.

  Στο πλαίσιο αυτής της κατηγορίας, αξίζει να αναφερθεί η προσέγγιση **Bottom-Up Agglomerative Approach** της παραπομπής [13]. Πρόκειται, μάλιστα, για μία παρόμοια προσέγγιση με αυτήν που χρησιμοποιήθηκε στο πλαίσιο του αλγορίθμου της ενότητας 2.4. Αρχικά, εντοπίζονται, με βάση τις μετρήσεις, οι δύο κόμβοι-προορισμοί (i, j) με την μεγαλύτερη εκτιμώμενη ομοιότητα ($\hat{\gamma}_{i,j}$). Κατά συνέπεια, οι δύο αυτοί κόμβοι βρίσκονται όσο πιο κοντά γίνεται στη λογική τοπολογία, δηλαδή έχουν κοινό "κόμβο-πατέρα" k. Έπειτα, οι i, j ομαδοποιούνται σε ένα σύνολο (**cluster**), το οποίο πλέον **αντιπροσωπεύεται από τον κόμβο k**. Στη συνέχεια, λαμβάνοντας υπόψη τον k που τώρα είναι ένας νέος κόμβος-προορισμός, υπολογίζονται οι **νέες ομοιότητες** μεταξύ εκείνου και των υπόλοιπων κόμβων της τοπολογίας. Παράλληλα, **ανακατασκευάζεται από κάτω προς τα πάνω το δέντρο της τοπολογίας**, θέτοντας τον k ως "πατέρα" των i, j. Επαναλαμβάνοντας τον συγκεκριμένο αλγόριθμο έως ότου απομείνει ένας κόμβος προς ομαδοποίηση, είναι δυνατή η ανακατασκευή ολόκληρης της τοπολογίας. Σημειώνεται ότι στο πρώτο στάδιο της μεθόδου η

εκτίμηση της ομοιότητας $\hat{\gamma}_{l,m}$ δύο κόμβων-προορισμών l, m ταυτίζεται με το στοιχείο (l, m) του πίνακα εκτιμώμενων ομοιοτήτων $\hat{\gamma}$. Ο πίνακας αυτός είναι εκείνος για τον οποίον μεγιστοποιείται η πιθανότητα $p(X|\gamma)$, όπου X: ο πίνακας των ¨εμπειρικών¨ ομοιοτήτων που προκύπτουν από τις μετρήσεις και γ: ο πίνακας των πραγματικών ομοιοτήτων.

Όσον αφορά τις **ανάγκες της εφαρμογής**, η οποία υλοποιείται στο πλαίσιο της παρούσας Διπλωματικής, ιδιαίτερης σημασίας είναι οι τομογραφικές μέθοδοι της πρώτης (**link-level parameter estimation**) και της τρίτης κατηγορίας (**topology inference**). Ο λόγος είναι ότι η συγκεκριμένη εφαρμογή επιχειρεί να εξάγει το πώς εξελίσσεται μία εσωτερική παράμετρος επίδοσης (απώλεια πακέτων) για κάθε ζεύξη της τοπολογίας. Με βάση αυτήν την πληροφορία σε επίπεδο ζεύξεων είναι δυνατή η ανίχνευση σημείων αλλαγής, δηλαδή σημείων στα οποία μεταβάλλεται η κατανομή, βάσει της οποίας η απώλεια πακέτων παίρνει τιμές σε κάθε ζεύξη. Συνεπώς, απαιτούνται ακραίες μετρήσεις σε επίπεδο μοναπατιού προκειμένου να εξαχθούν οι απαραίτητες πληροφορίες σε επίπεδο ζεύξης (link-level parameter estimation). Επίσης, δεδομένου, ότι ο τομογραφικός αλγόριθμος που επιστρατεύεται (ενότητα 2.4) ανακατασκευάζει και την τοπολογία του δικτύου (topology inference), η εφαρμογή άπτεται και της τρίτης κατηγορίας Τομογραφίας Δικτύου.

2.4 Αλγόριθμος συσσωρευτικής ομαδοποίησης

Για την υλοποίηση της εφαρμογής δικτυακής επιτήρησης στο πλαίσιο της παρούσας διπλωματικής εργασίας, επιλέχθηκε ο αλγόριθμος Τομογραφίας Δικτύου που παρουσιάζεται στην παραπομπή [8].

Με βάση το συγκεκριμένο αλγόριθμο, κάθε γράφος που αναπαριστά ένα δίκτυο μπορεί να γραφεί ως ένα σύνολο G = (V, E), όπου V: το σύνολο των κόμβων του και E: το σύνολο των ακμών (ζεύξεων) του. Σε κάθε ζεύξη του συνόλου E αντιστοιχεί μία παράμετρος επίδοσης ϑ_e, την οποία και επιχειρούμε να εκτιμήσουμε (μαζί με την τοπολογία του δικτύου) μέσω των ακραίων μετρήσεων. Για τον σκοπό αυτό ορίζουμε τις παραμέτρους Z_e και X_k, όπου Z_e: η μεταβλητή κατάστασης μιας ζεύξης $e \in E$ (link state variable) και X_k: η μεταβλητή εξόδου ενός κόμβου $k \in V$ (outcome variable for node k). Η δε μεταβλητή εξόδου X_k εξαρτάται από την αντίστοιχη μεταβλητή εξόδου του αμέσως προηγούμενου στην τοπολογία κόμβου σε σχέση με τον k (¨πατέρας¨ του k) ($X_{f(k)}$), καθώς και από τη μεταβλητή κατάστασης της ζεύξης μεταξύ εκείνου και του k (Z_{e_k}). Ακόμη d(e): το μήκος μιας ζεύξης $e \in E$ και d(i,j): η απόσταση μεταξύ των κόμβων i, j. Η κεντρική ιδέα είναι ότι **τόσο η τοπολογία όσο και οι αποστάσεις μεταξύ όλων των κόμβων μπορούν να καθοριστούν βάσει των αποστάσεων μεταξύ των τελικών κόμβων**[3].

Στην περίπτωση του συγκεκριμένου αλγορίθμου εξετάζονται **δύο δυνατές παράμετροι επίδοσης για κάθε ζεύξη**:

- Το ποσοστό απωλειών της ζεύξης (**loss**). Σε αυτήν την περίπτωση, η Z_e είναι μία τυχαία μεταβλητή Bernoulli, η οποία παίρνει δύο δυνατές τιμές: 1 σε περίπτωση που το πακέτο περνά από τη ζεύξη e (με πιθανότητα ϑ_e) και 0 σε περίπτωση που χαθεί στη ζεύξη (με πιθανότητα 1-ϑ_e). Αντίστοιχα, η X_k είναι μια τυχαία μεταβλητή Bernoulli, η οποία παίρνει τις τιμές: 1 όταν το πακέτο φτάνει στον κόμβο k και 0 όταν δεν φτάνει σε αυτόν, ενώ ορίζεται μέσω της σχέσης: $X_k = X_{f(k)} \cdot Z_{e_k}$

- Η χρονική καθυστέρηση της ζεύξης (**jitter**). Στην περίπτωση αυτή η Z_e είναι μία τυχαία μεταβλητή που εκφράζει την τυχαία καθυστέρηση λόγω ουράς πακέτων στη ζεύξη e. Αντίστοιχα, η X_k είναι μια μεταβλητή που εκφράζει την από άκρο σε άκρο (end-to-end) καθυστέρηση ενός πακέτου μέχρι να φτάσει στον κόμβο k, για την οποία ισχύει: $X_k = X_{f(k)} + Z_{e_k}$

[3] το σύνολο των κόμβων της τοπολογίας που περιλαμβάνει τον κόμβο-πηγή και τους κόμβους-προορισμούς

2.4.1 Προσδιορισμός πίνακα αποστάσεων

Με βάση τις **κατανομές των μεταβλητών εξόδου των τελικών κόμβων**, είναι δυνατό να υπολογιστεί ο πίνακας αποστάσεων (**distance matrix**) της τοπολογίας, δηλαδή ο πίνακας που περιέχει τις απόστασεις d(i,j) για κάθε συνδυασμό τελικών κόμβων i, j. Ωστόσο, επειδή στην πράξη οι **κατανομές** των μεταβλητών **δεν είναι γνωστές**, ο αλγόριθμος αξιοποιεί τις **ακραίες μετρήσεις**, προκειμένου να εξάγει **εκτιμήσεις** για τις εν λόγω αποστάσεις ($\hat{d}(i,j)$). Συγκεκριμένα, εάν μία πηγή s στείλει n πακέτα σε ένα σύνολο προορισμών D, οι μόνες μεταβλητές που μπορούν να μετρηθούν είναι οι μεταβλητές εξόδου (X_k) των τελικών κόμβων, δηλαδή οι: $X_k{}^{(t)} : k \in \{s\} \cup D$ και t: 1, 2, ..., n.

Έτσι, ανάλογα με το εάν μελετάται το ποσοστό απωλειών ή η χρονική καθυστέρηση της ζεύξης, ο πίνακας αποστάσεων υπολογίζεται ως εξής:

- Μελέτη ποσοστού απωλειών της ζεύξης:

$$\hat{d}(s,i) = log(\frac{1}{\overline{X_i}}) = -log(\overline{X_i}) \quad \forall i \in D \quad \text{και}$$

$$\hat{d}(i,j) = log(\frac{\overline{X_i} \cdot \overline{X_j}}{\overline{(X_i \cdot X_j)^2}}) \quad \forall i,j \in D$$

όπου $\overline{X_i} = \frac{1}{n} \cdot \sum_{t=1}^{n} X_i^{(t)}$ και $\overline{X_i X_j} = \frac{1}{n} \cdot \sum_{t=1}^{n} X_i^{(t)} X_j^{(t)}$

- Μελέτη καθυστέρησης της ζεύξης:

$$\hat{d}(s,i) = \widehat{Var}(X_i) \quad \forall i \in D \quad \text{και}$$

$$\hat{d}(i,j) = \widehat{Var}(X_i) + \widehat{Var}(X_j) - 2 \cdot \widehat{Cov}(X_i, X_j) \quad \forall i,j \in D$$

όπου $\widehat{Var}(X_i) = \frac{1}{n-1} \cdot \sum_{t=1}^{n} (X_i^{(t)} - \overline{X_i})^2$ και
$\widehat{Cov}(X_i, X_j) = \frac{1}{n-1} \cdot \sum_{t=1}^{n} (X_i^{(t)} - \overline{X_i}) \cdot (X_j^{(t)} - \overline{X_j})$

Σημειώνεται ότι, με βάση την παραπομπή [8], ο πίνακας αποστάσεων μπορεί να υπολογιστεί και μέσω μιας εναλλακτικής μεθόδου, η οποία βασίζεται στο βαθμό ανομοιότητας μεταξύ διαφορετικών δυαδικών ακολουθιών. Επειδή, ωστόσο, η μέθοδος αυτή δεν αξιοποιήθηκε στο πλαίσιο της παρούσας εφαρμογής, δεν θα αναλυθεί περαιτέρω.

2.4.2 Εξαγωγή λογικής τοπολογίας και παραμέτρων επίδοσης

Μετά τον υπολογισμό του πίνακα αποστάσεων ακολουθεί η εξαγωγή της λογικής τοπολογίας, καθώς και των παραμέτρων επίδοσης για κάθε ζεύξη της. Προκειμένου να επιτευχθεί κάτι τέτοιο, ορίζεται ο ¨ελάχιστος κοινός πρόγονος¨ (**least common ancestor - LCA**) μεταξύ δύο κόμβων i, j. Στην ουσία, πρόκειται για τον κόμβο που βρίσκεται μακρύτερα από την πηγή και έχει ως ¨παιδιά¨ του τους κόμβους i και j. Κατά συνέπεια, ορίζεται και το ¨βάθος του LCA¨ (**LCA depth**), το οποίο είναι η απόσταση του LCA από την πηγή και δίνεται από τη σχέση: $l(i,j) = d(s, LCA(i,j))$. Επομένως ισχύει ότι:

$$l(i,j) = \frac{1}{2} \cdot (d(s,i) + d(s,j) - d(i,j)), \quad \forall i,j \in D \quad \text{(II)} \quad \text{και}$$

$$l(i,i) = d(s,i) \quad \forall i \in D \quad \text{(III)}$$

Με βάση όλα τα παραπάνω, ο αλγόριθμος επιχειρεί να εξάγει την τοπολογία, ακολουθώντας τα εξής βήματα:

1. Αρχικά, υπολογίζει, με βάση τους τύπους (II) και (III), το LCA depth (l(i,j)) για κάθε συνδυασμό κόμβων i, j, δημιουργώντας τον πίνακα **LCA depth matrix**.

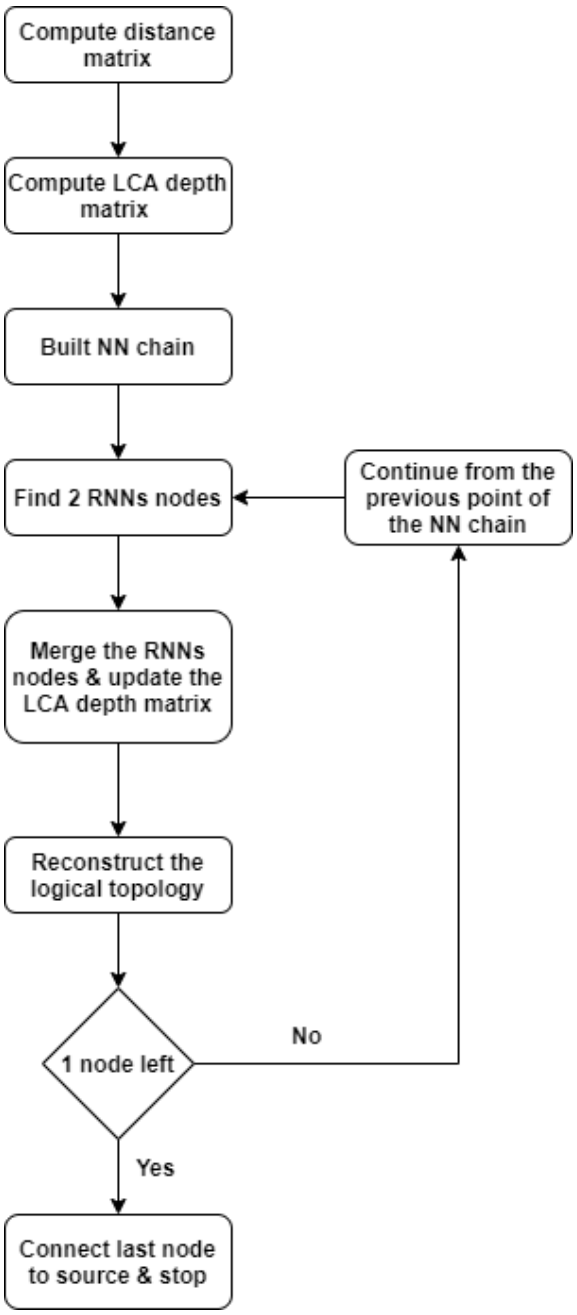
2. Στη συνέχεια, στόχος είναι να εντοπίσει τους κόμβους i', j' που έχουν το μεγαλύτερο LCA depth. Ωστόσο, επειδή κάτι τέτοιο είναι υπολογιστικά πολύπλοκο, ο αλγόριθμος εντοπίζει εναλλακτικά τους κόμβους i', j', για τους οποίους ισχύει ότι ο ένας είναι ο κοντινότερος γείτονας του άλλου (**Reciprocal Nearest Neighbors-RNNs**). Οι κόμβοι αυτοί προκύπτουν χαρτογραφώντας πάνω στην τοπολογία την αλυσίδα κόμβων Nearest Neighbors chain (**NN chain**). Συγκεκριμένα, επιλέγεται αρχικά ένας τυχαίος κόμβος i και, στη συνέχεια, ο κόμβος j, ο οποίος είναι ο κοντινότερος από άποψη LCA, δηλαδή $l(i,j) = max_{\mathrm{k} \neq \mathrm{i}} l(i,k)$. Ο κόμβος j είναι στην ουσία ο κοντινότερος γείτονας του i. Έπειτα η αλυσίδα NN chain επεκτείνεται προσαρτώντας τον κοντινότερο γείτονα του κόμβου j. Η διαδικασία αυτή επαναλαμβάνεται έως ότου βρεθούν δύο κόμβοι i', j', για τους οποίους ισχύει ότι ο ένας είναι ο κοντινότερος γείτονας του άλλου (RNNs κόμβοι).

3. Έπειτα, ο αλγόριθμος **ομαδοποιεί** τους κόμβους i', j' σε ένα νέο κόμβο u και **ανανεώνει**, με βάση αυτόν, **τις αποστάσεις του πίνακα LCA depth table** που αφορούν τους δύο κόμβους.

4. Τέλος, **ανακατασκευάζει την τοπολογία από κάτω προς τα πάνω**, προσθέτοντας τους κόμβους i', j' και τον κόμβο u ως πατέρα τους. Στη συνέχεια, ξεκινώντας από το σημείο της αλυσίδας NN chain που προηγούταν των i', j', η αλυσίδα επεκτείνεται πάλι έως ότου προκύψουν δύο νέοι RNNs κόμβοι, οι

οποίοι θα ομαδοποιηθούν με τον ίδιο τρόπο. Όταν δεν υπάρχουν πλέον διαθέσιμοι RNNs κόμβοι που να μπορούν να ομαδοποιηθούν και απομένει μόνο ένας κόμβος, ο τελευταίος αυτός κόμβος συνδέεται με την πηγή στην υπό ανακατασκευή τοπολογία και ο αλγόριθμος τερματίζεται.

Τα παραπάνω βήματα απεικονίζονται στο ακόλουθο διάγραμμα ροής.



Σχήμα 2.3: Διάγραμμα ροής αλγορίθμου συσσωρευτικής ομαδοποίησης

Κεφάλαιο 3

Offline Change Point Detection

3.1 Γενικά

Ο όρος "change point detection" αναφέρεται σε μία κατηγορία μεθόδων στατιστικής επεξεργασίας, οι οποίες εντοπίζουν σημεία αλλαγής (**change points**) σε ένα σήμα, δηλαδή σε μία χρονοσειρά (**time series**) [14]. Η τελευταία, στην ουσία, είναι ένα σύνολο τιμών που λαμβάνει ένα συγκεκριμένο μέγεθος σε βάθος χρόνου, διατεταγμένων κατά χρονολογική σειρά. Όσον αφορά τα σημεία αλλαγής, πρόκειται για τιμές του υπό μελέτη μεγέθους, οι οποίες αποκλίνουν αισθητά από τη μέση τιμή του, με βάση κάποιο καθορισμένο όριο απόκλισης (threshold) [15].

Οι **δύο βασικές υποκατηγορίες** "change point detection" είναι οι μέθοδοι **"online change point detection"**, οι οποίες συνίστανται στην ανίχνευση των σημείων αλλαγής σε πραγματικό χρόνο και οι **"offline change point detection"**, στις οποίες η ανίχνευση όλων των σημείων γίνεται εκ των υστέρων, δηλαδή αφότου ληφθούν όλες οι τιμές-δείγματα του εξεταζόμενου μεγέθους [14]. Στην παρούσα διπλωματική όλες οι μέθοδοι που χρησιμοποιήθηκαν ανήκουν στην υποκατηγορία "offline change point detection".

3.2 Ορισμός του προβλήματος

Στο σημείο αυτό παρουσιάζονται οι **βασικές έννοιες** των μεθόδων "change point detection", όπως αυτές ορίζονται στην παραπομπή [14].

Ένα σήμα ή μια χρονοσειρά μπορεί να αναπαρασταθεί ως ένα σύνολο T δειγμάτων $y = \{y_1, ..., y_T\} = \{y_t\}_{t=1}^{T}$. Αντίστοιχα, ένα υποσήμα του y στο διάστημα a έως b (όπου $1 \leq a < b \leq T$) συμβολίζεται ως $\{y_t\}_{t=a+1}^{b} = y_{a.b}$.

Δεδομένου ότι το σήμα εμφανίζει απότομες μεταβολές (δηλαδή σημεία αλλαγής) στα σημεία $t_1^*, t_2^*, ..., t_{K^*}^*$, οι μέθοδοι "change point detection" επιχειρούν να εκτιμήσουν

τα παραπάνω σημεία με όσο το δυνατόν μεγαλύτερη ακρίβεια. Στην ουσία, εξάγεται ένα σύνολο εκτιμώμενων σημείων αλλαγής $\mathcal{T} = \{t_1, t_2, ..., t_k\}$, με βάση το οποίο κατακερματίζεται το σήμα y. Στόχος είναι ο κατακερματισμός αυτός να προσεγγίζει όσο το δυνατό περισσότερο τον κατακερματισμό του σήματος, βάσει των πραγματικών σημείων αλλαγής $t_1^*, t_2^*, ..., t_{K^*}^*$. Ο δε υπολογισμός των εκτιμώμενων σημείων προκύπτει με βάση κάποια **συνάρτηση-κριτήριο**, όπως η ακόλουθη:

$$V(\mathcal{T}, y) = V(\mathcal{T}) = \sum_{k=0}^{K} c(y_{t_k.t_{k+1}})$$

όπου c(.): μία συνάρτηση κόστους (cost function), η οποία καθορίζει πόσο ¨ομοιογενές¨ είναι το υποσήμα $y_{t_k.t_{k+1}}$. Συγκεκριμένα, όσο μικρότερη είναι η τιμή που λαμβάνει η συνάρτηση κόστους για ένα δεδομένο υποσήμα, τόσο πιο ομογενές είναι αυτό, δηλαδή δεν περιέχει πολλά σημεία αλλαγής. Επομένως, η ελαχιστοποίηση της συνάρτησης $V(\mathcal{T})$ συνεπάγεται βέλτιστο κατακερματισμό του σήματος y με βάση τα προκύπτοντα σημεία αλλαγής. Κατ' επέκταση, η εκτίμηση των σημείων αλλαγής ενός σήματος ανάγεται στην επίλυση ενός προβλήματος βελτιστοποίησης, όπου στόχος είναι η ελαχιστοποίηση της $V(\mathcal{T})$.

Η εν λόγω εκτίμηση, μάλιστα, διαφοροποιείται ανάλογα με το εάν το πλήθος των πραγματικών σημείων αλλαγής (K^*) είναι γνωστό εκ των προτέρων ή όχι. Με βάση την παράμετρο αυτή, οι μέθοδοι "change point detection" διακρίνονται σε δύο βασικές κατηγορίες:

- **Γνωστό πλήθος (Κ)**: Στην περίπτωση αυτή, η εκτίμηση των σημείων αλλαγής είναι ακριβής και έγκειται στην ουσία στην επίλυση του προβλήματος βελτιστοποίησης:

  $$min_{|\mathcal{T}|=K} V(\mathcal{T}) \qquad \text{(I)}$$

- **Άγνωστο πλήθος**: Στην περίπτωση αυτή, η εκτίμηση των σημείων αλλαγής είναι προσεγγιστική και προκύπτει από την επίλυση του προβλήματος βελτιστοποίησης:

  $$min_{\mathcal{T}} V(\mathcal{T}) + pen(\mathcal{T}) \qquad \text{(II)}$$

  όπου $pen(\mathcal{T})$: παράμετρος, η οποία εκφράζει την πολυπλοκότητα του κατακερματισμού στα σημεία αλλαγής. Συγκεκριμένα, όσο μικρότερη είναι η τιμή της $pen(\mathcal{T})$, τόσο περισσότερα από τα άγνωστα σημεία αλλαγής ανιχνεύονται.

3.3 Βασικά στοιχεία

Συνεχίζοντας την ανάλυση της παραπομπής [14], κάθε μέθοδος "change point detection" μπορεί αναλυθεί σε τρία βασικά στοιχεία:

- Τη συνάρτηση κόστους (**cost function**), η οποία, όπως αναφέρθηκε και προηγουμένως, αποτελεί το μέτρο της ¨ομοιογένειας¨ ενός υποσήματος του αρχικού σήματος y.
- Τη μέθοδο αναζήτησης (**search method**), η οποία αποτελεί την αναλυτική διαδικασία που ακολουθείται για την επίλυση των προβλημάτων βελτιστοποίησης (I) και (II). Η μέθοδος αυτή μπορεί να είναι ακριβής ή προσεγγιστική, ανάλογα με τις εκάστοτε απαιτήσεις του προβλήματος.
- Τον περιορισμό (**constraint**) στον αριθμό των σημείων αλλαγής. Ο περιορισμός αυτός εισάγεται στην περίπτωση που ο αριθμός των πραγματικών σημείων αλλαγής (K^*) είναι άγνωστος (πρόβλημα βελτιστοποίησης (II)) και ταυτίζεται με την παράμετρο $pen(\mathcal{T})$. Όσο μικρότερη, δηλαδή, είναι η τιμή του, τόσο περισσότερα σημεία υπολογίζονται. Επομένως, το μέγεθος του περιορισμού εκφράζει την ¨ευαισθησία¨ κατά την εκτίμηση των σημείων αλλαγής.

3.4 Μοντέλα και κατηγορίες

Στη συνέχεια, και με βάση την παραπομπή [14], παρατίθενται οι βασικές κατηγορίες συναρτήσεων κόστους και μεθόδων αναζήτησης μαζί με ορισμένα αντιπροσωπευτικά παραδείγματα:

Συναρτήσεις κόστους (Cost functions):
Οι συναρτήσεις κόστους ομαδοποιούνται σε δύο βασικές κατηγορίες, αναλόγως το μοντέλο του σήματος y, το οποίο επιλέγεται κάθε φορά. Το μοντέλο αυτό μπορεί να είναι είτε **παραμετρικό** είτε **μη παραμετρικό**.

- Συναρτήσεις κόστους **παραμετρικών μοντέλων σήματος**:
  Στην περίπτωση των παραμετρικών μοντέλων, τα σημεία αλλαγής προκύπτουν από αλλαγές σε μία παράμετρο, η οποία αναπαρίσταται ως πίνακας πεπερασμένων διαστάσεων. Τα παραμετρικά μοντέλα σήματος, τα οποία συναντά κανείς στη βιβλιογραφία είναι τα εξής:

  1. **Maximum likelihood estimation**. Με βάση αυτό το μοντέλο, το σήμα $y = \{y_1, ..., y_T\}$ αποτελείται από ανεξάρτητες, τυχαίες μεταβλητές (i.i.d), έτσι ώστε να ισχύει:

$$y_t \approx \sum_{k=0}^{K^*} f(.|\theta_k) 1_A(t_k^* < t \leq t_{k+1}^*)$$

όπου t_k^*: τα σημεία αλλαγής και $f(.|\theta_k)$: η συνάρτηση πυκνότητας πιθανότητας της κατανομής των i.i.d μεταβλητών. Η δε συνάρτηση $f(.|\theta_k)$ εξαρτάται από τον πίνακα-παράμετρο θ του οποίου οι τιμές είναι οι θ_k. Στην ουσία η παράμετρος θ είναι αυτή που μεταβάλεται απότομα στα σημεία t_k^*, τα οποία και πρέπει να εκτιμηθούν.

Με βάση τα παραπάνω και ανάλογα με την εκάστοτε κατανομή των i.i.d μεταβλητών, προκύπτουν διάφορες συναρτήσεις κόστους, ορισμένες από τις οποίες είναι οι ακόλουθες:

(α') $\mathbf{c_{L2}}$: Η συνάρτηση αυτή χρησιμοποείται στην περίπτωση όπου οι i.i.d μεταβλητές ακολουθούν γκαουσιανή (Gaussian) κατανομή με σταθερή διακύμανση (Var) και μεταβαλλόμενη ανά διαστήματα μέση τιμή. Επομένως, η ανίχνευση των σημείων αλλαγής έγκειται στην ανίχνευση των σημείων μεταβολής της μέσης τιμής. Ο τύπος της συνάρτησης είναι ο εξής:

$$c_{L_2}(y_{a.b}) := \sum_{t=a+1}^{b} ||y_t - \overline{y}_{a.b}||_2^2$$

όπου $\overline{y}_{a.b}$: ο μέσος όρος των τιμών του υποσήματος $y_{a.b}$

(β') $\mathbf{c_{Poisson}}$: Η συνάρτηση αυτή αξιοποιείται όταν οι i.i.d μεταβλητές ακολουθούν κατανομή Poisson με μεταβαλλόμενη ανά διαστήματα παράμετρο λ και ορίζεται ως εξής:

$$c_{Poisson}(y_{a.b}) := -(b-a)\overline{y}_{a.b} log \overline{y}_{a.b}$$

2. **Multiple linear model**. Σύμφωνα με αυτό το μοντέλο, το σήμα y είναι γραμμικό ανά διαστήματα, επαληθεύοντας μια σχέση της μορφής:

$$y_t = x_t^{'} u_k + z_t^{'} v + \varepsilon_t \qquad k = (0, ..., K^*) \qquad \forall t, \quad t_k^* < t \leq t_{k+1}^*$$

η οποία μεταβάλλεται στα σημεία αλλαγής t_k^*. Στην εν λόγω σχέση, οι $u_k \in \Re^p$, $v \in \Re^q$ είναι άγνωστες παράμετροι ανάδρασης (regression parameters) και ε_t: ο θόρυβος. Επομένως, το σήμα y είναι μία μεταβλητή που παίρνει μονοδιάστατες τιμές (δηλαδή στον χώρο $\Re^d$ με $d = 1$) και τα $x = \{x\}_{t=1}^T$ και $z = \{z\}_{t=1}^T$ είναι συνιστώσες της που παίρνουν τιμές στον χώρο $\Re^p$ και $\Re^q$ αντίστοιχα.

Εφαρμόζοντας την παραπάνω ανάλυση για κάθε υποδιάστημα του σήματος μεταξύ δύο διαδοχικών σημείων αλλαγής, προκύπτουν οι ακόλουθες συναρτήσεις κόστους:

(α') **c_{linear}**: $c_{linear}(y_{a.b}) := min_{u \in \Re^p} \sum_{t=a+1}^{b} (y_t - x_t^{'} u)^2$

(β') **c_{L_1}**: $c_{L_1}(y_{a.b}) := min_{u \in \Re^p} \sum_{t=a+1}^{b} |y_t - x_t^{'} u - z_t^{'} v|$

Η συγκεκριμένη, μάλιστα, συνάρτηση κόστους προτιμάται στην περίπτωση όπου η γραφική αναπαράσταση της κατανομής του θορύβου ε_t εμφανίζει μεγάλες "ουρές".

3. **Mahalanobis-type metric** Με βάση το εν λόγω μοντέλο, το σήμα y ικανοποιεί την εξής σχέση:

$$||y_t||_M^2 := y_t^{'} M y_t$$

όπου $||.||_M$: η ημι-νόρμα[1] Mahalanobis-type seminorm και $M \in \Re^{dxd}$: ένας συμμετρικός ($M = M^T$), μη αρνητικά ορισμένος ($M \geq 0$)[2] πίνακας.

Η παραπάνω μοντελοποίηση οδηγεί στην επέκταση της συνάρτησης κόστους c_{L2} και στη διαμόρφωση της ακόλουθης συνάρτησης:

c_M: $c_M(y_{a.b}) := \sum_{t=a+1}^{b} ||y_t - \overline{y}_{a.b}||_M^2$

- Συναρτήσεις κόστους **μη παραμετρικών μοντέλων σήματος**:
Όταν δεν ικανοποιούνται οι προϋποθέσεις των παραμετρικών μοντέλων που περιγράφηκαν παραπάνω το σήμα y μοντελοποιείται ως εξής:

$$y_t \approx \sum_{k=0}^{K^*} F_k 1_A (t_k^* < t \leq t_{k+1}^*)$$

όπου οι τιμές του σήματος $\{y_1, ..., y_T\}$ θεωρούνται πάλι ανεξάρτητες, τυχαίες μεταβλητές (i.i.d) και F_k: η **συνάρτηση πυκνότητας πιθανότητας** των i.i.d μεταβλητών, η οποία, σε αντίθεση με πριν, **είτε δεν είναι παραμετρική, είτε δεν είναι γνωστή εκ των προτέρων**.

[1] γενίκευση της νόρμας διανυσμάτων, ώστε μη μηδενικά διανύσματα να μπορούν να έχουν μηδενικό μέτρο [16]

[2] δηλαδή όλες οι ιδιοτιμές του είναι μη αρνητικές

Εφαρμόζοντας την εν λόγω μοντελοποίηση, προκύπτουν οι ακόλουθες αντιπροσωπευτικές συναρτήσεις κόστους:

c_{rank}: Προκειμένου να εφαρμοστεί η εν λόγω συνάρτηση, απαιτείται η αντιστοίχιση του σήματος y σε ένα σήμα διαβαθμίσεων (rank signal), το οποίο ορίζεται ως εξής:

$$r_{t,j} := \sum_{s=1}^{T} 1_A(y_{s,j} \leq y_{t,j}) - \frac{T+1}{2}, \quad \forall 1 \leq t \leq T \quad \forall 1 \leq j \leq d$$

όπου $r_{t,j}$: η διαβάθμιση της j-οστής διάστασης (*θεωρώντας ότι το y παίρνει τιμές στον χώρο* $\Re^d$) του t-οστού δείγματος y_t.

Κατά συνέπεια, η συνάρτηση κόστους είναι η ακόλουθη:

$$c_{rank}(y_{a.b}) := -(b-a)\overline{r}'_{a.b}\widehat{\sum}_r^{-1}\overline{r}_{a.b}$$

όπου $\overline{r}_{a.b}$: ο μέσος όρος των τιμών του υποσήματος $r_{a.b}$ και $\widehat{\sum}_r \in \Re^{dxd}$ ο πίνακας $\widehat{\sum}_r := \frac{1}{T}\sum_{t=1}^{T}(r_t + \frac{1_d}{2})(r_t + \frac{1_d}{2})'$ ($1_d \in \Re^d$: πίνακας του οποίου όλες οι τιμές του είναι ίσες με 1)

c_{rbf}: Για να εφαρμοστεί η εν λόγω συνάρτηση απαιτείται η αντιστοίχιση των σημείων του σήματος y σε έναν αναπαραγόμενο χώρο Hilbert (Hilbert reproducing space) H, ο οποίος με τη σειρά του σχετίζεται με μία καθορισμένη από τον χρήστη συνάρτηση γκαουσιανού πυρήνα (Gaussian kernel function) $k : \Re^d \times \Re^d \to \Re$. Η συνάρτηση δε μέσω της οποίας πραγματοποείται η αντιστοίχιση είναι η $\phi : \Re^d \to H$, ενώ η συνάρτηση γκαουσιανού πυρήνα δίνεται από τον τύπο: $k(x,y) = exp(-\gamma||x-y||^2)$, όπου $x, y \in \Re^d$ και $\gamma > 0$: η παράμετρος εύρους ζώνης.

Με βάση τα παραπάνω, η συνάρτηση κόστους είναι η εξής:

$$c_{rbf}(y_{a.b}) := (b-a) - \frac{1}{b-a} \cdot \sum_{s,t=a+1}^{b} exp(-\gamma||y_s - y_t||^2)$$

Μέθοδοι αναζήτησης (Search methods):

Οι μέθοδοι αναζήτησης ταξινομούνται σε δύο βασικές κατηγορίες, ανάλογα με το εάν οδηγούν σε ακριβή ή προσεγγιστική λύση των προβλημάτων βελτιστοποίησης (I) και (II). Συγκεκριμένα οι κατηγορίες είναι οι εξής:

- **Βέλτιστες Μέθοδοι (Optimal Methods)**: Οι μέθοδοι αναζήτησης αυτής της κατηγορίας δίνουν ακριβή λύση και στα δύο προβλήματα βελτιστοποίησης. Σε καθένα, μάλιστα, από τα δύο προβλήματα αντιστοιχούν οι εξής βέλτιστες μέθοδοι αναζήτησης:

  - **Πρόβλημα βελτιστοποίησης (I)**: Η μέθοδος αναζήτησης, η οποία επιστρατεύεται για την ακριβή επίλυση αυτού του προβλήματος είναι η **μέθοδος Opt**. Η μέθοδος αυτή υλοποιεί ανδρομικά, με τη χρήση **δυναμικού προγραμματισμού**, την εξής σχέση:

    $$min_{|\mathcal{T}|=K} V(\mathcal{T}, y) = min_{t \leq T-K}[c(y_{0.t}) + min_{|\mathcal{T}|=K-1} V(\mathcal{T}, y_{t.T})]$$

    Δηλαδή, το πρώτο από τα K σημεία αλλαγής του βέλτιστου κατακερματισμού του y ($min_{|\mathcal{T}|=K} V(\mathcal{T}, y)$) προκύπτει εάν υπολογιστεί ο βέλτιστος κατακερματισμός όλων των υποσημάτων $y_{t.T}$ για K-1 σημεία αλλαγής. Εφαρμόζοντας, λοιπόν, αναδρομικά την παραπάνω λογική, είναι δυνατός ο υπολογισμός του βέλτιστου κατακερματισμού ολόκληρου του σήματος y.

  - **Πρόβλημα βελτιστοποίησης (II)**: Η ακριβής μέθοδος αναζήτησης που αντιστοιχεί σε αυτό το πρόβλημα είναι η **μέθοδος Pelt (Pruned Exact Linear Time)**. Προκειμένου, ωστόσο, να εφαρμοστεί η συγκεκριμένη μέθοδος, θα πρέπει η παράμετρος $pen(\mathcal{T})$ να είναι γραμμική, δηλαδή: $pen(\mathcal{T}) = \beta|\mathcal{T}|$, όπου $\beta > 0$: μία παράμετρος εξομάλυνσης. Στη συνέχεια, όλες οι τιμές του σήματος $y = \{y_1, ..., y_T\}$ εξετάζονται διαδοχικά και κάθε μία συμπεριλαμβάνεται ή απορρίπτεται από το σύνολο ενδεχόμενων σημείων αλλαγής, βάσει ενός κανόνα (**pruning rule**). Ο κανόνας αυτός δίνεται παρακάτω:

    Για δύο δείκτες τιμών του σήματος y, t και s ($t < s < T$) ισχύει ότι:

    $$\text{εάν} \quad [min_{\mathcal{T}} V(\mathcal{T}, y_{0.t}) + \beta|\mathcal{T}|] + c(y_{t.s}) \geq [min_{\mathcal{T}} V(\mathcal{T}, y_{0.s}) + \beta|\mathcal{T}|]$$

    τότε ο δείκτης t δεν μπορεί να είναι το τελευταίο σημείο αλλαγής πριν τον δείκτη T.

Ο παραπάνω έλεγχος έχει ως αποτέλεσμα να μειώνεται σημαντικά ο απαιτούμενος χρόνος εύρεσης των σημείων αλλαγής, ενώ η πολυπλοκότητα του αντίστοιχου αλγορίθμου είναι της τάξης O(T).

- **Προσεγγιστικές Μέθοδοι (Approximate Methods)**: Οι μέθοδοι αναζήτησης που εμπίπτουν σε αυτήν την κατηγορία αποτελούν μία εναλλακτική λύση στην περίπτωση που η υπολογιστική πολυπλοκότητα των βέλτιστων μεθόδων είναι απαγορευτική. Τρεις αντιπροσωπευτικές προσεγγιστικές μέθοδοι που συναντά κανείς στη βιβλιογραφία είναι οι ακόλουθες:

  - **Ολίσθηση παραθύρου (Window sliding-Win)**: Κατά την υλοποίηση αυτής της μεθόδου, εξετάζεται κάθε φορά η ¨ανομοιότητα¨ μεταξύ δύο γειτονικών διαστημάτων του σήματος y, καθώς αυτά μετακινούνται (¨ολισθαίνουν¨) κατά μήκος του. Με αυτόν τον τρόπο διαμορφώνεται μία καμπύλη ανομοιότητας, της οποίας τα σημεία προκύπτουν από την ανομοιότητα των εκάστοτε γειτονικών διαστημάτων. Όταν δε η ανομοιότητα αυτή είναι αρκετά μεγάλη τότε στο αντίστοιχο σημείο της καμπύλης παρουσιάζεται μία κορυφή (peak). Στη συνέχεια, κι αφού έχει σχηματιστεί ολόκληρη η καμπύλη ανομοιότητας, πραγματοποιείται μία αναζήτηση των κορυφών της και κάθε κορυφή αντιστοιχίζεται σε ένα σημείο αλλαγής.

  - **Δυαδικός κατακερματισμός (Binary segmentation-BinSeg)**: Στην περίπτωση αυτής της μεθόδου, πραγματοποιείται, αρχικά, αναζήτηση του σημείου εκείνου, για το οποίο ελαχιστοποιείται το άθροισμα των συναρτήσεων κόστους των δύο υποσημάτων που ορίζονται με βάση το σημείο αυτό. Το ζητούμενο σημείο t δίνεται, δηλαδή, από τον τύπο:

    $$\hat{t}^{(1)} := argmin_{1 \leq t < T-1} \quad c(y_{0.t}) + c(y_{t.T})$$

    Μετά τον εντοπισμό του, το σημείο αυτό αποτελεί το πρώτο σημείο αλλαγής του σήματος και η διαδικασία επαναλαμβάνεται διαδοχικά στα υποσήματα που προκύπτουν κάθε φορά, έως ότου ικανοποιηθεί ένα κριτήριο τερματισμού.

– **Από κάτω προς τα πάνω κατακερματισμός (Bottom-up segmentation-BotUp)**: Η μέθοδος αυτή αποτελεί στην ουσία τη συμπληρωματική μέθοδο της Binary segmentation-BinSeg. Συγκεκριμένα, το σήμα, αρχικά, διαιρείται σε μικρότερα υποσήματα, τα οποία, στη συνέχεια, ομαδοποιούνται, έως ότου προκύψουν K σημεία αλλαγής. Η ομαδοποίηση αυτή πραγματοποιείται σε βήματα ως εξής: Σε κάθε βήμα, όλα τα πιθανά σημεία αλλαγής, δηλαδή όλα τα σημεία τα οποία ορίζουν γειτονικά υποσήματα, ταξινομούνται με βάση την ανομοιότητα των υποσημάτων που ορίζει το καθένα. Τα σημεία, για τα οποία προκύπτει η μικρότερη ανομοιότητα διαγράφονται από τα πιθανά σημεία αλλαγής και τα υποσήματα, τα οποία ορίζουν ομαδοποιούνται.

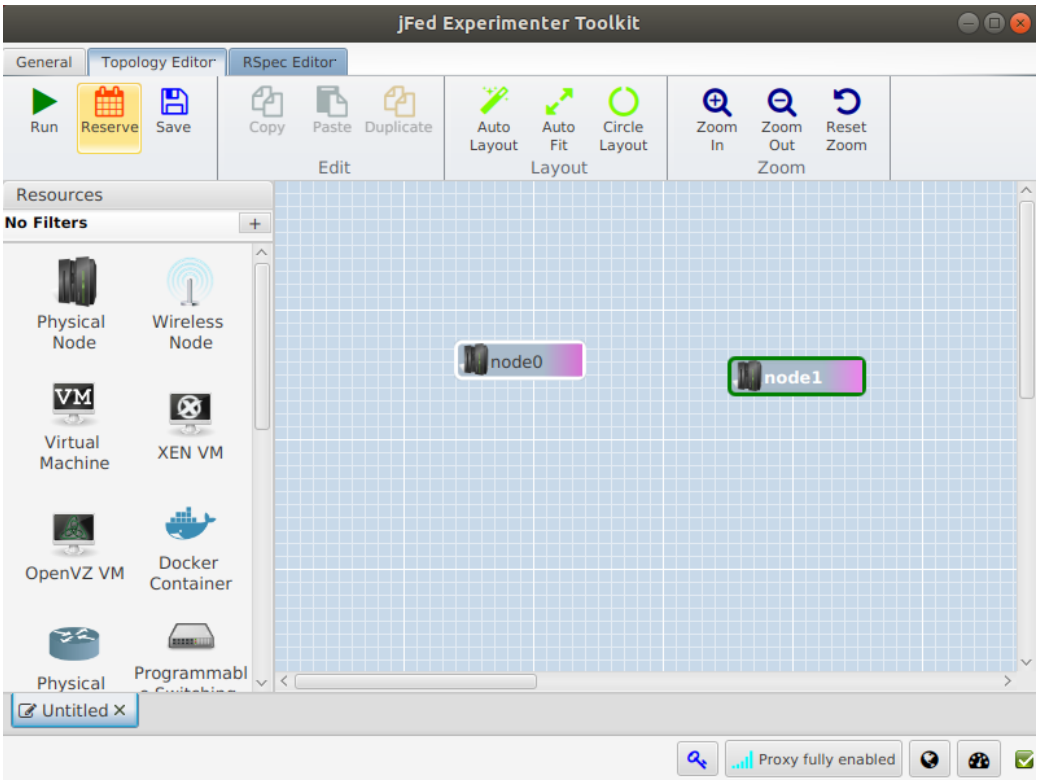
Κεφάλαιο 4

Πειραματική Διάταξη και Εργαλεία

4.1 Κατασκευή τοπολογίας μέσω του jFed

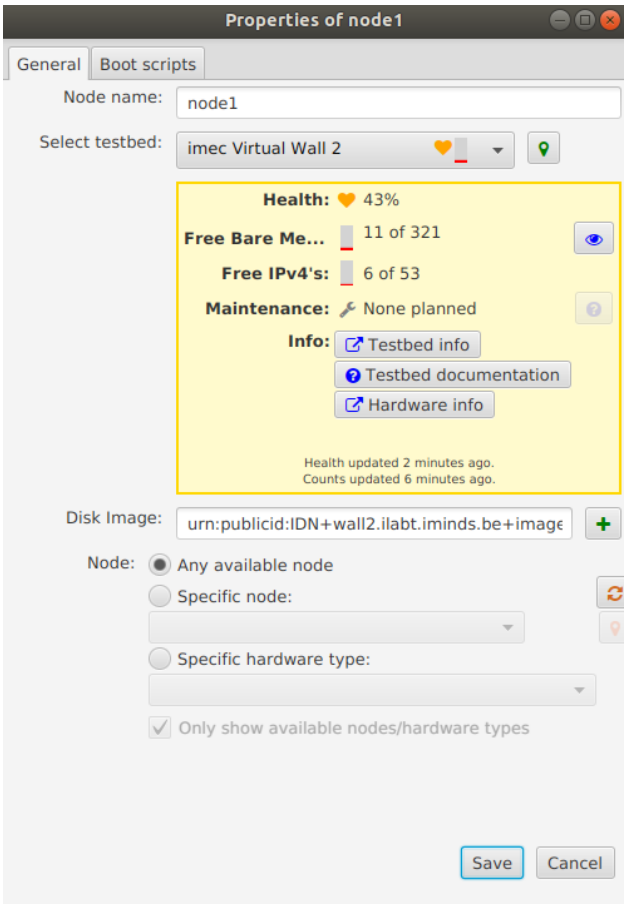
Για την υλοποίηση της εφαρμογής δικτυακής επιτήρησης **κατασκευάστηκε μία πρότυπη τοπολογία μέσω του jFed**. Το τελευταίο αποτελεί, στην ουσία, ένα σύνολο εργαλείων (framework), υλοποιημένο σε Java, το οποίο επιτρέπει τη χρήση ενός δικτύου πλατφορμών δοκιμών (testbed federation) για τη διεξαγωγή πειραμάτων. Τόσο το jFed, όσο και το αντίστοιχο testbed federation, ανήκουν στο project Fed4FIRE. Αξίζει, επίσης, να σημειωθεί ότι το jFed είναι συμβατό με την αρχιτεκτονική διαχείρισης SFA (Slice-based Federation Architecture), ενώ για τη χρήση των εργαλείων του παρέχει διεπαφή γραμμής εντολών (CLI), αλλά και γραφική διεπαφή χρήστη (GUI) [7].

Κατά τη δημιουργία μίας πειραματικής διάταξης μέσω του jFed, ο χρήστης **ορίζει αρχικά τους κόμβους της τοπολογίας**, επιλέγοντας μέσα από ένα μενού διαφορετικών μηχανημάτων (για παράδειγμα φυσικοί ή ασύρματοι κόμβοι, δρομολογητές ή μεταγωγείς) και προσθέτοντας τους, μέσω drag and drop. Τα παραπάνω απεικονίζονται στο σχήμα 4.1 της επόμενης σελίδας.



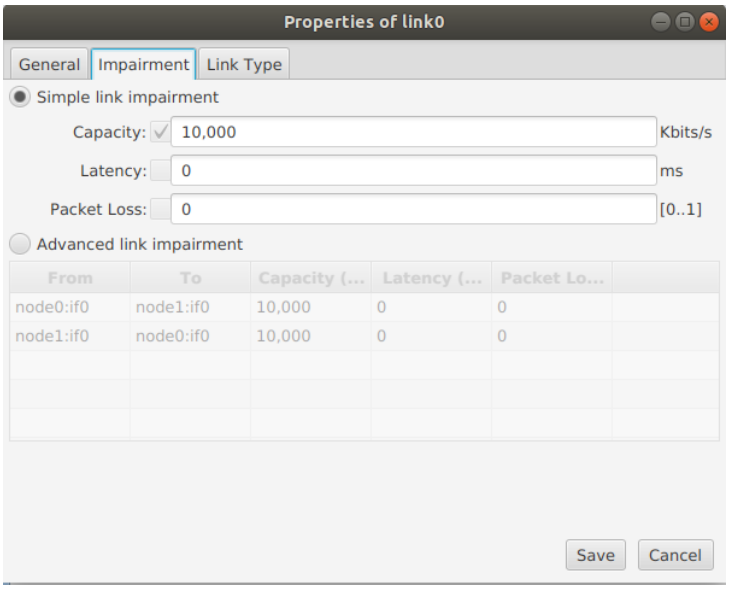
Σχήμα 4.1: Εισαγωγή μηχανημάτων στο jFed

Στη συνέχεια, **παραμετροποιεί τους κόμβους της τοπολογίας**, καθορίζοντας το λειτουργικό σύστημα που υποστηρίζουν τα αντίστοιχα μηχανήματα, μέσω του πεδίου Disk Image. Παράλληλα, έχει τη δυνατότητα να επιλέξει και το testbed από το οποίο θα δεσμευθεί το κάθε μηχάνημα, μέσω του πεδίου Select testbed, όπως φαίνεται και από το σχήμα 4.2.



Σχήμα 4.2: Παραμετροποίηση μηχανημάτων στο jFed

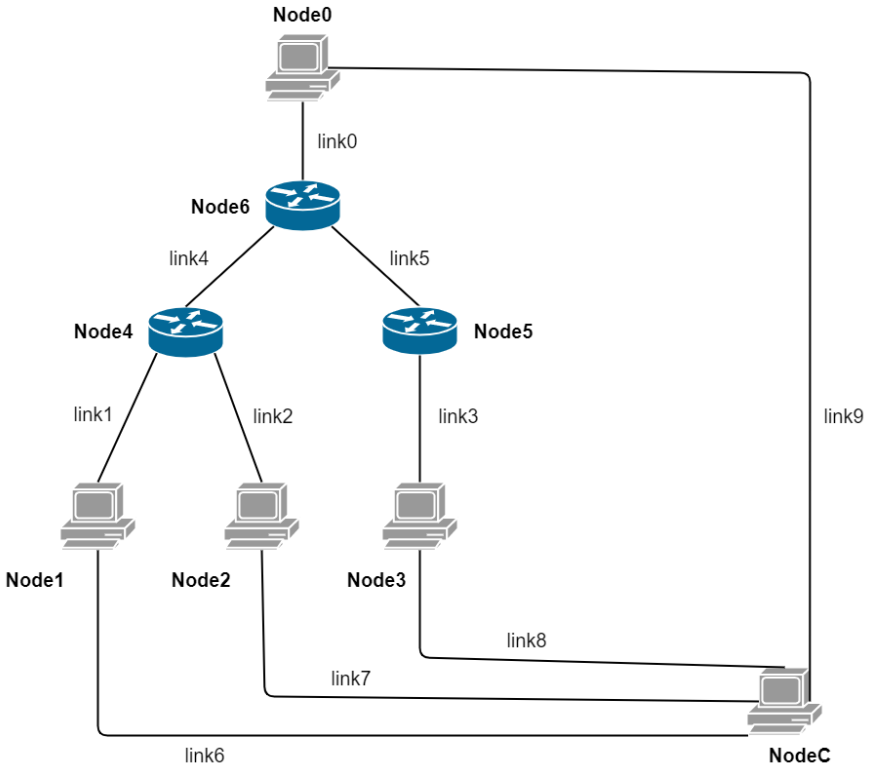
Έπειτα, ενώνει τους κατάλληλους κόμβους μεταξύ τους, ώστε να **σχηματίσει τις ζητούμενες ζεύξεις της τοπολογίας**, τις οποίες, μάλιστα, μπορεί να ρυθμίσει ως προς την καθυστέρηση τους, αλλά και το ποσοστό απώλειας πακέτων τους. Οι ρυθμίσεις αυτές πραγματοποιούνται μέσω της καρτέλας Impairment και από εκεί μέσω των πεδίων Latency και Packet Loss αντίστοιχα (σχήμα 4.3).



Σχήμα 4.3: Παραμετροποίηση ζεύξεων στο jFed

Όταν η τοπολογία έχει σχηματιστεί και παραμετροποιηθεί πλήρως ο χρήστης, πατώντας το κουμπί Run, εκκινεί το πείραμα. Συγκεκριμένα, το jFed **δεσμεύει τους πόρους** που έχει ορίσει ο χρήστης από τα αντίστοιχα testbeds και μόλις αυτοί είναι διαθέσιμοι προχωρά στην επιβεβαίωσή τους (**configuration**), ώστε να μπορούν μετέπειτα να είναι προσβάσιμοι από εκείνον.

Στο πλαίσιο της παρούσας διπλωματικής εργασίας, επιλέχθηκαν ως κόμβοι της τοπολογίας τα εξής **8 μηχανήματα** με λειτουργικό σύστημα Linux Ubuntu 18.04 LTS: **1** κόμβος-πηγή (**source**) (**Node0**), **3** εσωτερικοί κόμβοι-δρομολογητές (**routers**) (**Node4, Node5, Node6**), **3** κόμβοι-προορισμοί (**clients**) (**Node1, Node2, Node3**) και **1** κόμβος για τη διαχείριση της ροής πακέτων και της εκτέλεσης της εφαρμογής (**controller**) (**NodeC**). Όλα τα μηχανήματα δεσμεύτηκαν από το testbed imec Virtual Wall 1, ενώ η διασύνδεσή τους είναι ορατή στην ακόλουθη εικόνα.



Σχήμα 4.4: Τοπολογία εφαρμογής στο jFed

4.2 Quagga

Για τη δρομολόγηση των πακέτων από τους δρομολογητές Node4, Node5 και Node6 χρησιμοποιήθηκε η σουίτα λογισμικού δρομολόγησης (**routing software suite**) Quagga. Η συγκεκριμένη σουίτα παρέχει τη δυνατότητα εφαρμογής πολλαπλών πρωτοκόλλων δρομολόγησης, όπως το RIP, το OSPF και το BGP σε λειτουργικά συστήματα βασισμένα στο Unix [17]. Στο πλαίσιο της συγκεκριμένης διπλωματικής εργασίας, η δρομολόγηση της δικτυακής κίνησης από τους δρομολογητές Node4, Node5 και Node6 έγινε μέσω του πρωτοκόλλου **OSPF (Open Shortest Path First)** για τις **unicast εκπομπές** (ospfd daemon) και του **PIM (Protocol Independent Multicast)** για τις **multicast εκπομπές** (pimd daemon) [18]. Επίσης, αξιοποιήθηκε ο διαχειριστής δρομολόγησης IP (IP routing manager) Zebra, ο οποίος παρέχει ενημερώσεις του πίνακα δρομολόγησης του πυρήνα (kernel routing table updates), αναζητήσεις διεπαφών (interface lookups) και αναδιανομή των διαδρομών δρομολόγησης μεταξύ των διαφορετικών πρωτοκόλλων (redistribution of routes between different routing protocols). Τέλος, επιστρατεύτηκε η ενσωματωμένη διεπαφή φλοιού (integrated shell) του Quagga, vtysh [19].

4.3 Iperf-ssm και tcpdump

Όσον αφορά τη διαδικασία αποστολής πακέτων και λήψης μετρήσεων, χρησιμοποιήθηκε τόσο στον κόμβο-πηγή (Node0), όσο και στους κόμβους-προορισμούς (Node1, Node2, Node3), το εργαλείο iperf-ssm. Πρόκειται για ένα εργαλείο, το οποίο χρησιμοποιείται για τη μέτρηση της απόδοσης δικτύων [20]. Συγκεκριμένα, καθ' όλη τη διάρκεια εκτέλεσης της εφαρμογής, ο **Node0** λειτουργούσε ως **iperf client**, στέλνοντας multicast πακέτα με πρωτόκολλο στρώματος μεταφοράς UDP, ενώ οι **Node1, Node2 και Node3** λειτουργούσαν ως **iperf servers**, οι οποίοι λάμβαναν τα παραπάνω πακέτα.

Η δε καταγραφή των πακέτων που αποστέλλονταν από τον κόμβο-πηγή, αλλά και των πακέτων που λαμβάνονταν από τους κόμβους-προορισμούς, πραγματοποιούταν μέσω του εργαλείου tcpdump. Το συγκεκριμένο εργαλείο παρέχει τη δυνατότητα απεικόνισης των πακέτων που εισέρχονται ή εξέρχονται από μία διεπαφή δικτύου, χρησιμοποιώντας μία λογική έκφραση-κανόνα. Όσα πακέτα πληρούν αυτήν τη λογική έκφραση απεικονίζονται μαζί με τις σχετικές πληροφορίες τους, όπως για παράδειγμα η διεύθυνση πηγής και προορισμού [21]. Στην προκειμένη περίπτωση, η λογική έκφραση καθόριζε τη διεπαφή στην οποία γινόταν η καταγραφή, το πρωτόκολλο στρώματος μεταφοράς των ζητούμενων πακέτων (UDP), τη θύρα προέλευσης ή προορισμού, την αναπαράσταση της πληροφορίας, καθώς και το ότι τα λαμβανόμενα πακέτα θα έπρεπε να αποθηκεύονται σε ένα αρχείο txt. Για παράδειγμα η λογική έκφραση για τη διεπαφή δικτύου του Node1 ήταν η 'sudo tcpdump

-nlvvv -i enp8s0f1 udp and port 5001 | tee DATA1' + str(i+1) + '.txt'. Στην εντολή αυτή η παράμετρος -nlvvv υποδηλώνει ότι οι διευθύνσεις των πακέτων δεν θα μεταφράζονται στα αντίστοιχα domain names (-n), ενώ οι πληροφορίες των πακέτων θα απεικονίζονται σε πραγματικό χρόνο (-l) και όσο το δυνατόν πιο αναλυτικά (-vvv). Αντίστοιχα η παράμετρος -i enp8s0f1 ορίζει τη διεπαφή καταγραφής, η παράμετρος udp το πρωτόκολλο στρώματος μεταφοράς των πακέτων και η παράμετρος port 5001 τη θύρα προέλευσης τους. Τέλος, λόγω της επιλογής '| tee DATA1' + str(i+1) + '.txt', τα καταγραφόμενα πακέτα αποθηκεύονται σε ένα αρχείο DATA1(i+1).txt, όπου i: ο αύξων αριθμός της καταγραφής.

4.4 Iptables

Σχετικά με την εφαρμογή της απώλειας πακέτων στις ζεύξεις της τοπολογίας, αξιοποιήθηκε το πρόγραμμα γραμμής εντολών (command line program) iptables. Το εν λόγω πρόγραμμα επιτρέπει τη σύνταξη ενός συνόλου κανόνων, με βάση τους οποίους ¨φιλτράρονται¨ τα πακέτα του δικτύου. Όταν, δηλαδή, ένα πακέτο πληροί κάποιον κανόνα, τότε εφαρμόζεται η ενέργεια που ο κανόνας αυτός περιγράφει. Με αυτό τον τρόπο δημιουργείται στην ουσία ένα firewall, το οποίο εξετάζει και κατευθύνει τη δικτυακή κίνηση, βασιζόμενο σε κριτήρια όπως η διεύθυνση προορισμού [22], [23]. Στην παρούσα εφαρμογή δικτυακής επιτήρησης, το iptables χρησιμοποιήθηκε για τη σύνταξη κανόνων, οι οποίοι εφαρμόστηκαν από τους **δρομολογητές Node4, Node5 και Node6**. Συγκεκριμένα, εάν τα πακέτα που έφταναν σε αυτούς ήταν **multicast εκπομπής**, τότε, ανάλογα με τη διεπαφή εξόδου στην οποία προωθούνταν, **απορρίπτονταν και με συγκεκριμένη πιθανότητα**. Η πιθανότητα αυτή ήταν η πιθανότητα απώλειας πακέτων που είχε οριστεί για τη ζεύξη της εκάστοτε διεπαφής εξόδου. Για παράδειγμα, τα multicast πακέτα που έφταναν στον Node4 και προωθούνταν στη διεπαφή εξόδου του στη ζεύξη link1 απορρίπτονταν με πιθανότητα ίση με την πιθανότητα απώλειας πακέτων στη ζεύξη link1. Η εντολή, μέσω της οποίας επιτυγχανόταν αυτό, ήταν η 'sudo iptables -A FORWARD -d 239.1.2.3 -o enp8s0f0 -m statistic –mode random –probability ' + links41 + ' -j DROP'. Στη συγκεκριμένη εντολή, η παράμετρος -A FORWARD υποδηλώνει την προσθήκη του εν λόγω κανόνα στο σύνολο κανόνων, οι οποίοι αφορούν τα πακέτα που προωθούνται (FORWARD chain). Αντίστοιχα, η παράμετρος -d 239.1.2.3 καθορίζει τη διεύθυνση προορισμού των multicast πακέτων, ενώ η παράμετρος -o enp8s0f0 τη διεπαφή από την οποία ο Node4 τα προωθεί στη ζεύξη link1. Τέλος, η επιλογή '–probability ' + links41 + ' -j DROP' δηλώνει ότι τα πακέτα τα οποία εμπίπτουν στον εν λόγω κανόνα θα απορρίπτονται (-j DROP) με πιθανότητα ίση με την πιθανότητα απώλειας πακέτων στη ζεύξη link1 (links41).

Σημειώνεται ότι ο ορισμός της απώλειας πακέτων στις ζεύξεις θα μπορούσε εναλλα-

κτικά να πραγματοποιηθεί μέσω του εργαλείου γραμμής εντολών (command line tool) tc. Πρόκειται, στην ουσία, για ένα εργαλείο μέσω του οποίου ο χρήστης ελέγχει το πρόγραμμα netem που είναι εγκατεστημένο στον πυρήνα των Linux. Το πρόγραμμα αυτό δίνει τη δυνατότητα τροποποίησης της σειράς με την οποία δρομολογούνται τα πακέτα από τον πυρήνα. Ωστόσο, οι κανόνες δρομολόγησης που ορίζονται από τον χρήστη εφαρμόζονται σε όλα τα πακέτα ανεξαιρέτως. Επομένως, σε περίπτωση που η απώλεια πακέτων στις ζεύξεις εφαρμοζόταν μέσω του tc, θα υπήρχε η πιθανότητα απόρριψης πακέτων πέρα από τα ζητούμενα multicast πακέτα, όπως για παράδειγμα πακέτων συγχρονισμού μεταξύ των κόμβων [24], [25].

4.5 ZeroMQ

Για τον συγχρονισμό και την ανταλλαγή πληροφοριών μεταξύ των κόμβων του δικτύου ενσωματώθηκε στον κώδικά τους η βιβλιοθήκη ασύγχρονης δικτυακής αποστολής μηνυμάτων (**asynchronous network messaging library**) ZeroMQ. Το ZeroMQ δίνει τη δυνατότητα στα επιμέρους στοιχεία ενός δικτύου να επικοινωνούν μεταξύ τους, ανταλλάσσοντας μηνύματα μέσω δομών, οι οποίες ονομάζονται **sockets** κι ακολουθώντας παράλληλα κάποιο ¨μοτίβο¨ (**pattern**) επικοινωνίας. Για κάθε δύο, μάλιστα, στοιχεία, τα οποία πρόκειται να επικοινωνήσουν μεταξύ τους, δημιουργείται ένα ίδιο socket και στα δύο. Για παράδειγμα, στο πλαίσιο της συγκεκριμένης εφαρμογής, η επικοινωνία μεταξύ του NodeC και καθενός από τους υπόλοιπους κόμβους πραγματοποιούταν μέσω ενός ξεχωριστού κάθε φορά socket, το οποίο είχε δημιουργηθεί και σε εκείνον και στον εκάστοτε κόμβο.

Όσον αφορά το μοτίβο επικοινωνίας που επιλέγεται κάθε φορά οι διαθέσιμες επιλογές είναι οι εξής: Η πρώτη είναι το μοτίβο **request-reply**, σύμφωνα με το οποίο σε ένα ζεύγος επικοινωνίας το ένα στοιχείο (client) ξεκινά, κάνοντας ένα αίτημα μέσω του socket του (REQ socket), ενώ το άλλο (server) ξεκινά, περιμένοντας να λάβει ένα αίτημα μέσω του socket του (REP socket). Στη συνέχεια, οι λειτουργίες των δύο αντιστρέφονται και η διαδικασία αυτή επαναλαμβάνεται για όσες φορές χρειαστεί, χωρίς ωστόσο να μπορεί να αλλάξει η σειρά των ρόλων του καθενός.

Η δεύτερη επιλογή είναι το μοτίβο **publish-subscribe**, το οποίο χρησιμοποιείται για μονόδρομη διάδοση πληροφορίας. Στην ουσία, ένα στοιχείο επικοινωνίας (publisher) ενημερώνει τακτικά μέσω του socket του (PUB socket) ένα σύνολο άλλων στοιχείων (subscribers), προωθώντας δεδομένα στα sockets τους (SUB sockets). Κάθε SUB socket έχει μία ή περισσότερες ¨συνδρομές¨ (subscriptions), οι οποίες αποθηκεύονται σαν ένα σύνολο. Εάν κάποια από τις ενημερώσεις του publisher εμπίπτει σε αυτό το σύνολο, τότε το SUB socket την λαμβάνει. Αναφορικά με τη σειρά των ρόλων κάθε στοιχείου, ο publisher ξεκινά κάνοντας ένα αίτημα, χωρίς να

μπορεί ποτέ να λάβει και οι subscribers το αντίστροφο.

Η τρίτη επιλογή είναι το μοντέλο **pipeline**, το οποίο χρησιμοποιείται για την παράλληλη εκτέλεση ενεργειών και τη συλλογή των αντίστοιχων αποτελεσμάτων. Συγκεκριμένα, ένα στοιχείο επικοινωνίας (ventilator) στέλνει μέσω του socket του (PUSH socket) καθήκοντα προς εκτέλεση σε ένα σύνολο άλλων στοιχείων (workers). Οι workers λαμβάνουν την πληροφορία, μέσω του ενός από τα δύο sockets που διαθέτουν (PULL socket) και εκτελούν τις ανάλογες ενέργειες, οι οποίες μπορούν να εκτελεστούν παράλληλα από τους workers. Στη συνέχεια, οι workers προωθούν τα αποτελέσματά τους μέσω του δεύτερου socket τους (PUSH socket) και τα αποτελέσματα αυτά συλλέγονται από ένα άλλο στοιχείο επικοινωνίας (sink), μέσω του PULL socket που διαθέτει.

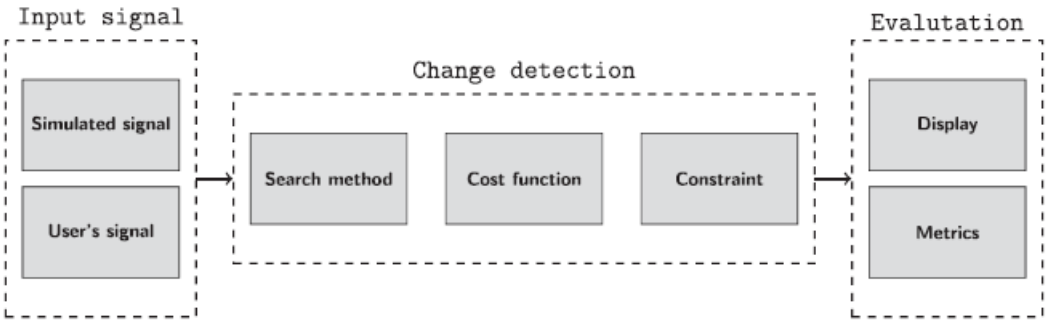
Λαμβάνοντας υπόψη τα παραπάνω, το μοτίβο επικοινωνίας το οποίο επιλέχθηκε στο πλαίσιο της **συγκεκριμένης εφαρμογής** είναι το **request-reply**. Αυτό οφείλεται στο ότι, με βάση τη λειτουργία της εφαρμογής, ο controller (NodeC) έπρεπε όχι μόνο να ενημερώνει τους κόμβους Node0, Node1, Node2, Node3, προκειμένου να εκκινήσουν τις λειτουργίες τους, αλλά να ενημερώνεται και ο ίδιος από εκείνους όταν τις ολοκλήρωναν. Επομένως, ήταν απαραίτητη η υλοποίηση μιας αμφίδρομης ροής πληροφορίας. Ακόμη, οι ενέργειες που έπρεπε να εκτελεστούν από κάθε κόμβο ήταν απαραίτητο να εκτελούνται διαδοχικά κι όχι παράλληλα, ώστε να διασφαλίζεται ότι καμία αποστολή πακέτων δεν θα ξεκινούσε αν δεν είχε εκκινήσει προηγουμένως η κατάλληλη καταγραφή. Γι' αυτούς τους λόγους, το καταλληλότερο μοντέλο επικοινωνίας ήταν το request-reply [26]–[28].

4.6 Ruptures

Τέλος, για την εφαρμογή μεθόδων offline change point detection πάνω στα αποτελέσματα του αλγορίθμου Τομογραφίας Δικτύου, χρησιμοποιήθηκε η βιβλιοθήκη ruptures της Python. Πρόκειται για μία βιβλιοθήκη εξ' ολοκλήρου υλοποιημένη σε γλώσσα Python, διαθέσιμη για τα λειτουργικά συστήματα Mac OS X, Linux και Windows. Η δομή της είναι βασισμένη στις **3 βασικές παραμέτρους** που καθορίζουν τις μεθόδους offline change point detection και αναλύθηκαν στην ενότητα 3.4. Συγκεκριμένα, η βιβλιοθήκη παρέχει στον χρήστη ένα σύνολο διαφορετικών **συναρτήσεων κόστους**, από τις οποίες ορισμένες βασίζονται σε παραμετρικά μοντέλα, ενώ άλλες σε μη παραμετρικά. Αντίστοιχα, δίνεται η δυνατότητα επιλογής μεταξύ βέλτιστων και προσεγγιστικών **μεθόδων αναζήτησης**, ενώ, σε περίπτωση που ο αριθμός των εκτιμώμενων σημείων αλλαγής δεν είναι εξαρχής γνωστός, μπορεί να επιλεχθεί κάποιος **περιορισμός**. Όσον αφορά την είσοδο που αναγνωρίζει η συγκεκριμένη βιβλιοθήκη, προκειμένου να εφαρμόσει την εκάστοτε μέθοδο offline

change point detection, αυτή μπορεί να είναι είτε κάποιο σήμα (δηλαδή χρονοσειρά) που παράγεται μέσω δικής της συνάρτησης, είτε κάποιο σήμα καθορισμένο από τον χρήστη. Στη δεύτερη περίπτωση, ο χρήστης μπορεί να εισάγει ως είσοδο οποιοδήποτε σήμα (μίας ή περισσότερων διαστάσεων), το οποίο έχει αναπαρασταθεί ως πίνακας Numpy. Η αναπαράσταση δε των εκτιμήσεων για τα σημεία αλλαγής του σήματος-χρονοσειράς πραγματοποιείται μέσω ενός γραφήματος, του οποίου ο οριζόντιος άξονας αναπαριστά τα δείγματα του σήματος και ο κάθετος τις αντίστοιχες τιμές τους. Στο γράφημα αυτό η εναλλαγή των δύο χρωμάτων υποδηλώνει την ύπαρξη ενός πραγματικού σημείου αλλαγής σε εκείνο το σημείο, ενώ η διακεκομμένη γραμμή την εκτίμησή του. Ακόμη, δίνεται η δυνατότητα στο χρήστη να εφαρμόσει ορισμένες μετρικές αξιολόγησης (evaluation metrics), μέσω των οποίων μπορεί να συγκρίνει ποσοτικά την ακρίβεια με την οποία οι διαφορετικές μέθοδοι κατακερματίζουν το σήμα με βάση τα εκτιμώμενα σημεία αλλαγής [14].

Τόσο η δομή, όσο και ο τρόπος λειτουργίας της βιβλιοθήκης συνοψίζονται στο ακόλουθο σχήμα:



Σχήμα 4.5: Δομή και λειτουργία ruptures, [14]

Κεφάλαιο 5

Αλγόριθμος και Υλοποίηση Εφαρμογής

Κατά την εκτέλεση της εφαρμογής δικτυακής επιτήρησης εκτελείται ο ακόλουθος αλγόριθμος. Ο συγκεκριμένος αλγόριθμος μπορεί να εφαρμοστεί σε οποιαδήποτε δενδρική τοπολογία.

Αρχικά, ο controller (στην προκειμένη περίπτωση ο NodeC) ορίζει **για κάθε ζεύξη** της τοπολογίας **μία μέση τιμή απώλειας πακέτων**, η οποία επιλέγεται με ισοπίθανο τρόπο από ένα προκαθορισμένο διάστημα τιμών. Εξαίρεση αποτελεί η ζεύξη που συνδέεται με τον κόμβο-πηγή για την οποία δεν ορίζεται μέση τιμή (link0). Στη συνέχεια, **για κάθε μία από τις επαναλήψεις** που έχει ορίσει εκ των προτέρων ο χρήστης ακολουθούνται **τα παρακάτω στάδια**.

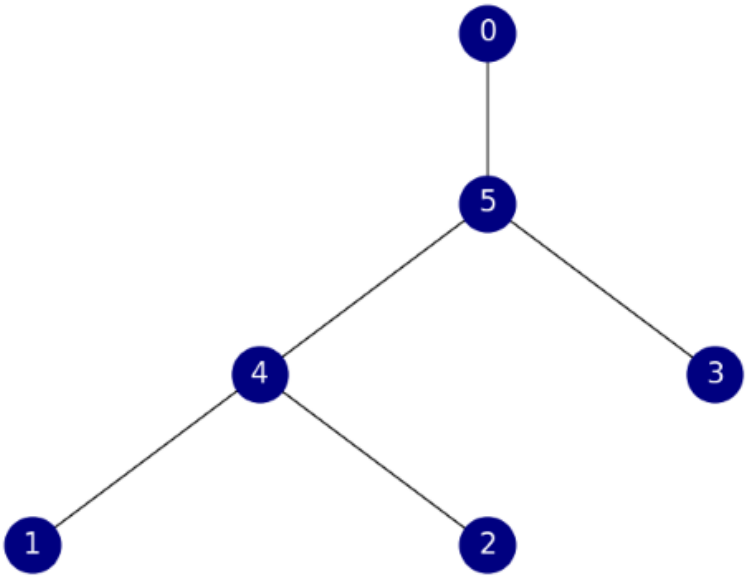
- Αρχικά, με βάση τις προηγούμενες μέσες τιμές, εξάγεται η τιμή της απώλειας πακέτων (**loss rate**) που θα ανατεθεί **σε κάθε ζεύξη**. Συγκεκριμένα, για κάθε μία από τις μέσες τιμές πραγματοποιείται δειγματοληψία μέσω της κατανομής Βήτα (Beta distribution) και με μέση τιμή δειγματοληψίας την εν λόγω μέση τιμή. Στη συνέχεια, κάθε μία από τις προκύπτουσες τιμές απώλειας αποστέλλεται διαδοχικά από τον controller στον αντίστοιχο εσωτερικό κόμβο, ο οποίος με τη σειρά του στέλνει ένα μήνυμα επιβεβαίωσης πίσω στον controller για κάθε τιμή που λαμβάνει. Για παράδειγμα, οι τιμές απώλειας για τις ζεύξεις link1 (ή αλλιώς 4 — 1) και link2 (4 — 2) αποστέλλονται στον Node4, ο οποίος επιστρέφει το αντίστοιχο μήνυμα επιβεβαίωσης για κάθε μία από αυτές. Σημειώνεται ότι ο controller στέλνει την επόμενη τιμή απώλειας ζεύξης σε έναν κόμβο μόνο αφότου έχει λάβει μήνυμα επιβεβαίωσης για την προηγούμενη τιμή που έστειλε.

- Αφού ο controller λάβει μήνυμα επιβεβαίωσης από όλους τους κόμβους, ξεκινά να στέλνει διαδοχικά στους κόμβους-προορισμούς (Node1, Node2, Node3) ένα ¨μήνυμα εκκίνησης¨ (**initialization message**). Καθένας από τους κόμβους-προορισμούς μόλις λάβει το συγκεκριμένο ¨μήνυμα εκκίνησης¨ ξεκινά αρχικά

την καταγραφή πακέτων στη διεπαφή στην οποία περιμένει να λάβει multicast πακέτα από την κόμβο-πηγή (Node0) (για παράδειγμα ο Node1 στη διεπαφή του στη ζεύξη 4 — 1). Αμέσως μετά μπαίνει σε ρόλο iperf server, αναμένοντας multicast πακέτα από τον κόμβο-πηγή. Και σε αυτό το στάδιο (όπως και σε όλα τα επόμενα) ακολουθείται η ίδια λογική μηνυμάτων επιβεβαίωσης που ακολουθήθηκε και στο προηγούμενο στάδιο. Αφού, λοιπόν, κάθε κόμβος-προορισμός έχει εκκινήσει τις δύο παραπάνω λειτουργίες, ο controller στέλνει ¨μήνυμα εκκίνησης¨ στον κόμβο-πηγή, έτσι ώστε κι εκείνος, με τη σειρά του, να ξεκινήσει πρώτα την καταγραφή πακέτων στην κατάλληλη διεπαφή (αυτή της ζεύξης 0 — 6) κι έπειτα την αποστολή multicast πακέτων στο δίκτυο ως iperf client. Διευκρινίζεται ότι οι καταγραφές κάθε κόμβου πραγματοποιούνται μέσω του tcpdump και αποθηκεύονται σε ένα ξεχωριστό αρχείο txt για κάθε επανάληψη.

- Όταν ο κόμβος-πηγή ολοκληρώσει την αποστολή multicast πακέτων, σταματά την καταγραφή μέσω του tcpdump και στέλνει το σχετικό αρχείο txt στον controller. Ο τελευταίος **ενημερώνει διαδοχικά τους κόμβους-προορισμούς**, ώστε αρχικά να σταματήσουν τη λήψη πακέτων και, στη συνέχεια, να ολοκληρώσουν την καταγραφή, στέλνοντας πίσω σε αυτόν το αρχείο txt τους.

- Έχοντας λάβει όλα τα αρχεία καταγραφής, ο controller **εκτελεί τον κώδικα Τομογραφίας Δικτύου** που παρουσιάστηκε στην ενότητα 2.4, εξάγοντας έτσι για τη δεδομένη επανάληψη τις εκτιμήσεις της απώλειας πακέτων σε κάθε ζεύξη, καθώς και την αντίστοιχη λογική τοπολογία του δικτύου. Οι εν λόγω εκτιμήσεις προστίθενται σε ένα αρχείο csv μαζί με τις εκτιμήσεις των προηγούμενων επαναλήψεων. Σημειώνεται ότι οι ζεύξεις για τις οποίες πραγματοποιούνται οι εκτιμήσεις είναι οι ζεύξεις της εξαγόμενης λογικής τοπολογίας (σχήμα 5.1). Αυτός είναι και ο λόγος που, στην προκειμένη περίπτωση, η απώλεια πακέτων στη λογική ζεύξη 5 — 3 υπολογίζεται από τον συνδυασμό των απωλειών στις φυσικές ζεύξεις 6 — 5 και 5 — 3 της τοπολογίας του jFed. Επίσης, λόγω της φύσης του αλγορίθμου, η εκτίμηση για τη ζεύξη της λογικής τοπολογίας, η οποία συνδέεται με τον κόμβο-πηγή (0 — 5) είναι σε κάθε επανάληψη ίση με μηδέν. Γι' αυτόν τον λόγο, οι εκτιμήσεις που αφορούν τη συγκεκριμένη ζεύξη απορρίπτονται μετά το πέρας των επαναλήψεων και δεν χρησιμοποιούνται στην περαιτέρω ανάλυση.



Σχήμα 5.1: Λογική τοπολογία του σχήματος 4.4

Εκτός, όμως, από τον αριθμό των επαναλήψεων, **ο χρήστης έχει τη δυνατότητα να θέσει και ορισμένες από τις επαναλήψεις αυτές ως σημεία αλλαγής (change points)**, δηλαδή επαναλήψεις στις οποίες η απώλεια πακέτων μίας ή περισσότερων ζεύξεων μεταβάλλεται αισθητά σε σχέση με τις προηγούμενες τιμές της. Στις επαναλήψεις αυτές προτού εξαχθεί η τιμή της απώλειας πακέτων για κάποια από τις συγκεκριμένες ζεύξεις, επιλέγεται νέα μέση τιμή από το προκαθορισμένο διάστημα, η οποία και διατηρείται για τις επόμενες επαναλήψεις. Με αυτόν τον τρόπο, οι τιμές που θα εξάγονται πλέον θα κυμαίνονται γύρω από διαφορετική μέση τιμή και άρα θα διαφέρουν σημαντικά από τις προηγούμενες.

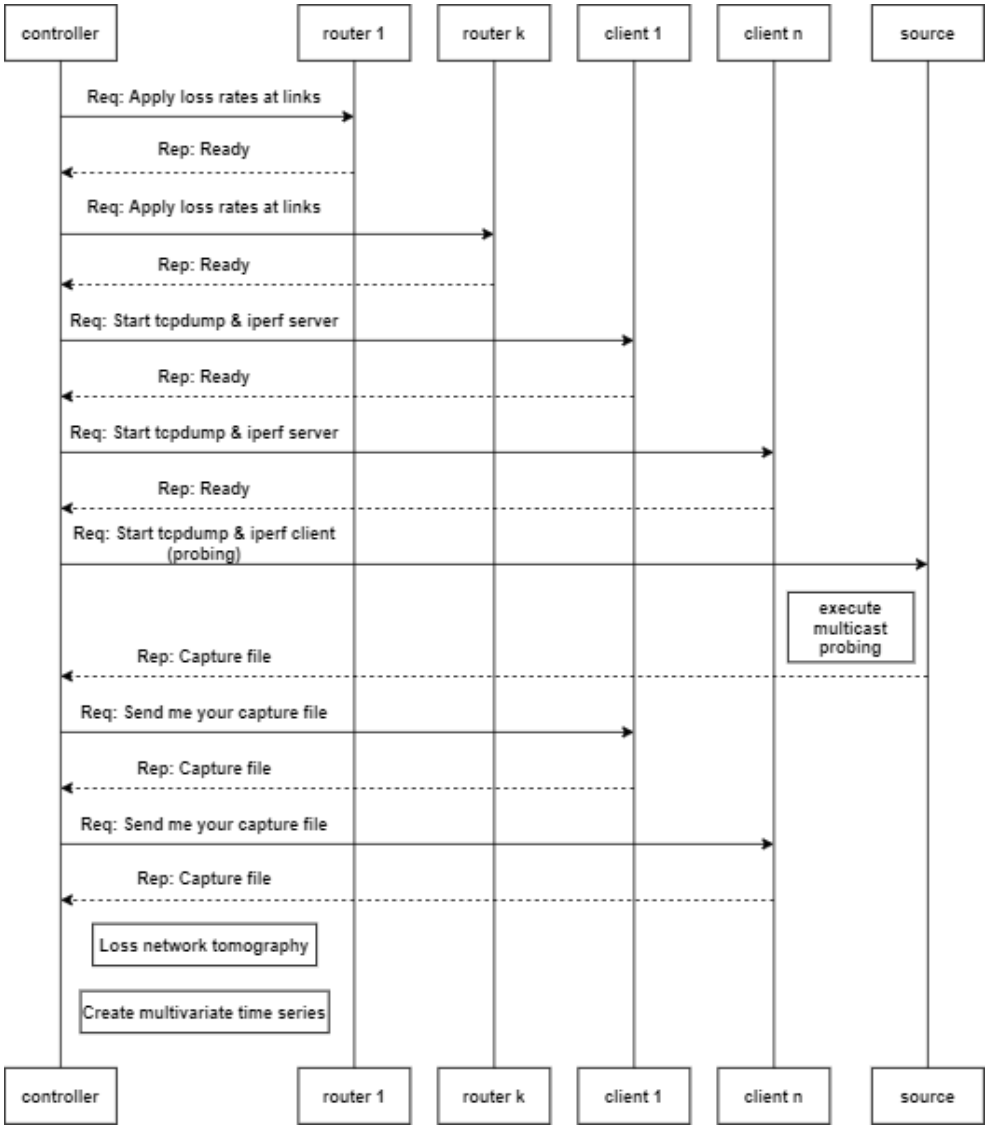
Όταν ολοκληρωθούν όλες οι επαναλήψεις, το τελικό αρχείο csv μετατρέπεται σε ένα DataFrame (σχήμα 5.2), από το οποίο εξαιρούνται τα μηδενικά δεδομένα της ζεύξης που συνδέεται με τον κόμβο-πηγή. Στη συνέχεια, το τελικό DataFrame αξιοποιείται για **στατιστική ανάλυση τύπου offline change point detection**. Για τον σκοπό αυτό, χρησιμοποιείται η βιβλιοθήκη ruptures της Python, χάρη στην οποία τα δεδομένα του DataFrame μπορούν να αντιμετωπιστούν ως μία πολυδιάστατη χρονοσειρά (multivariate time series), κάθε διάσταση της οποίας αντιστοιχεί σε μία από τις ζεύξεις του DataFrame (4 — 1, 4 — 2, 5 — 3 και 5 — 4). Το τελικό αποτέλεσμα είναι τα εκτιμώμενα σημεία αλλαγής (change points), συνοδευόμενα από ένα γράφημα τεσσάρων υπογραφημάτων, όπου το καθένα αντιστοιχεί σε μία από τις παραπάνω 4 ζεύξεις.

	4 --- 1	4 --- 2	5 --- 3	5 --- 4
0	0.11921	0.06651	0.15740	0.09976
1	0.13190	0.07464	0.15367	0.10310
2	0.11513	0.07179	0.15839	0.10142
3	0.12050	0.07395	0.14939	0.10656
4	0.11720	0.08255	0.15434	0.09631
..	...	...	...	...
95	0.10907	0.09177	0.15585	0.05935
96	0.10307	0.09749	0.14778	0.04825
97	0.09091	0.09459	0.15425	0.06298
98	0.11411	0.09066	0.15471	0.05601
99	0.10695	0.09177	0.15949	0.06043

Σχήμα 5.2: DataFrame καταγραφής εκτιμήσεων

Μεταβάλλοντας, μάλιστα, τις τρεις βασικές παραμέτρους της offline change point detection, οι οποίες αναλύθηκαν στην ενότητα 3.4 (cost function, search method, και constraint), μπορεί κανείς να εξετάσει την ακρίβεια κάθε συνδυασμού ως προς την εκτίμηση των σημείων αλλαγής. Η **συγκριτική μελέτη** ορισμένων από τους συνδυασμούς αυτούς παρουσιάζεται αναλυτικά στο επόμενο κεφάλαιο.

Στο ακόλουθο διάγραμμα παρουσιάζεται συνοπτικά η σειρά με την οποία πραγματοποιούνται οι διάφορες αλληλεπιδράσεις μεταξύ του controller και των υπόλοιπων κόμβων του σχήματος 4.1. Οι δε κώδικες σε Python που εκτελούνται από κάθε κόμβο είναι διαθέσιμοι στο Παράρτημα.



Σχήμα 5.3: Διάγραμμα ακολουθίας

Κεφάλαιο 6

Αποτελέσματα και Συγκριτική Μελέτη

6.1 Γενικά

Στο κεφάλαιο αυτό, αρχικά, παρατίθενται τα αποτελέσματα, τα οποία προέκυψαν από την εκτέλεση της εφαρμογής δικτυακής επιτήρησης (ενότητα 6.2). Για λόγους αντικειμενικότητας, θεωρήθηκε ότι **το πλήθος των σημείων αλλαγής δεν είναι εξαρχής γνωστό** και κατ' επέκταση χρησιμοποιήθηκε ένας περιορισμός (constraint), μέσω της παραμέτρου pen, την οποία παρέχει η βιβλιοθήκη ruptures.

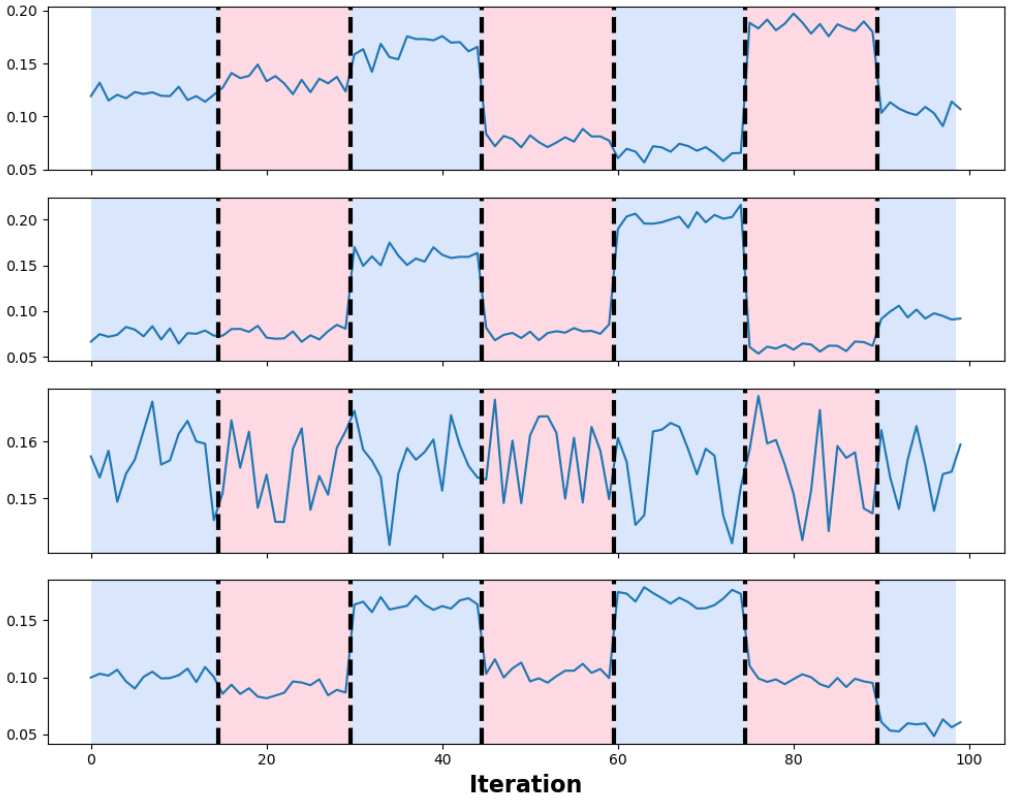
Ωστόσο, μετά την παραπάνω εκτέλεση, το csv αρχείο, το οποίο παράχθηκε από αυτήν χρησιμοποιήθηκε προκειμένου να δοκιμαστούν κι άλλες μέθοδοι offline change point detection. Για τον σκοπό αυτό, εκτελέστηκε τοπικά σε ηλεκτρονικό υπολογιστή το κομμάτι του κώδικα του controller (NodeC), το οποίο αφορούσε την στατιστική επεξεργασία των αποτελεσμάτων, εισάγοντας κάθε φορά ως είσοδο το εν λόγω αρχείο csv και **μεταβάλλοντας τη μέθοδο offline change point detection**. Στην ουσία, μεταβαλλόταν κάθε φορά μία ή περισσότερες από τις τρεις βασικές παραμέτρους της ενότητας 3.4 (cost function, search method, και constraint). Σκοπός ήταν (για το ίδιο σύνολο δεδομένων) να μελετηθεί συγκριτικά η ακρίβεια των διαφορετικών συνδυασμών, ως προς την εκτίμηση των σημείων αλλαγής.

Ο τρόπος με τον οποίον επιτεύχθηκε αυτό είναι ο εξής: **Για κάθε συνδυασμό** βρισκόταν **ένα εύρος τιμών** του περιορισμού (pen) για το οποίο εκτιμώνταν σωστά όλα τα σημεία αλλαγής. Οι τιμές αυτές ήταν τέτοιες, ώστε, αφενός, να μην υπολογίζονται (λόγω υπερβολικά μικρού περιορισμού) περισσότερα σημεία απ' όσα υπήρχαν (false positive change points) κι αφετέρου, να μην παραλείπονται σημεία, λόγω υπερβολικής "ανοχής" κατά την εκτίμηση. Στη συνέχεια, τα προκύπτοντα εύρη τιμών **συγκρίνονταν ως προς το άνω όριό τους**. Όσο μεγαλύτερο ήταν το όριο αυτό, τόση μικρότερη "ευαισθησία" απαιτείτο από τον αντίστοιχο συνδυασμό, ώστε να γίνει σωστή εκτίμηση κι άρα τόσο πιο ακριβής ήταν ο συνδυασμός. Σε περίπτωση που για

κάποια μέθοδο αναζήτησης υπήρχε συνάρτηση κόστους, η οποία δεν εκτιμούσε σωστά όλα τα σημεία αλλαγής, η σύγκριση περιοριζόταν στις υπόλοιπες συναρτήσεις. Τέλος, εάν για κάποια μέθοδο αναζήτησης καμία συνάρτηση κόστους δεν εκτιμούσε σωστά όλα τα σημεία αλλαγής, τότε επιλεγόταν ως βέλτιστη η συνάρτηση που επέφερε την καλύτερη προσέγγιση και με τη μικρότερη απαιτούμενη ¨ευαισθησία¨. Τα αποτελέσματα της παραπάνω συγκριτικής μελέτης παρατίθενται στην ενότητα 6.2.

6.2 Αποτελέσματα

Κατά την **πρωταρχική εκτέλεση** της εφαρμογής υπήρξαν οι ακόλουθες παραμετροποιήσεις στους κώδικες των κόμβων. Αρχικά, ο αριθμός των επαναλήψεων, οι οποίες αναφέρθηκαν στο κεφάλαιο 5, ήταν 100, προκειμένου να υπάρξει ένα επαρκές δείγμα πάνω στο οποίο θα μπορούσε να εφαρμοστεί στατιστική ανάλυση μέσω offline change point detection. Από αυτές τις **100 επαναλήψεις** εκείνες, οι οποίες επιλέχθηκαν ως **σημεία αλλαγής**, ήταν οι επαναλήψεις **15, 30, 45, 60, 75 και 90**. Όσον αφορά την ίδια τη στατιστική ανάλυση, η συνάρτηση κόστους, η οποία εφαρμόστηκε μέσω της βιβλιοθήκης ruptures, ήταν η Kernelized mean change (c_{rbf}), ενώ η μέθοδος αναζήτησης ήταν η Pelt. Τέλος, ο περιορισμός που εφαρμόστηκε μέσω της παραμέτρου pen ήταν ίσος με 0.3, ώστε να έχουμε όσο το δυνατόν ακριβέστερη εκτίμηση. Τα αποτελέσματα που προέκυψαν παρουσιάζονται στο ακόλουθο γράφημα, ενώ οι κώδικες των κόμβων είναι διαθέσιμοι στο Παράρτημα.



Σχήμα 6.1: Pelt, c_{rbf}, pen=0.3

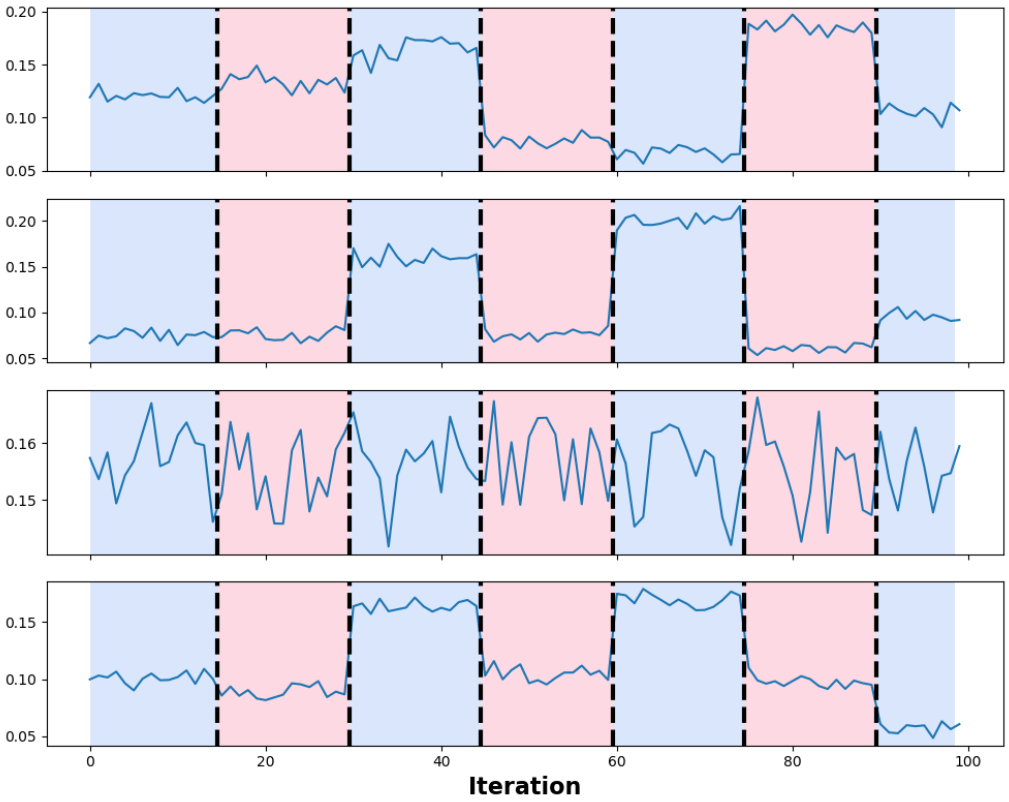
Στην παραπάνω απεικόνιση το πρώτο (από πάνω προς τα κάτω) υπογράφημα αντιστοιχεί στη ζεύξη 4 — 1 της λογικής τοπολογίας του σχήματος 5.1. Το δεύτερο αντιστοιχεί στη ζεύξη 4 — 2, το τρίτο στη ζεύξη 5 — 3 και το τέταρτο στη ζεύξη 5 — 4. Παρατηρεί κανείς ότι οι εναλλαγές των χρωμάτων (πραγματικά σημεία αλλαγής) συμπίπτουν απόλυτα με τις διακεκομμένες γραμμές (εκτιμώμενα σημεία αλλαγής) κι επομένως όλα τα σημεία (15, 30, 45, 60, 75, 90) ανιχνεύονται με ακρίβεια.

6.3 Συγκριτική μελέτη

Οι συνδυασμοί μεθόδου αναζήτησης και συνάρτησης κόστους που δοκιμάστηκαν είναι οι εξής:

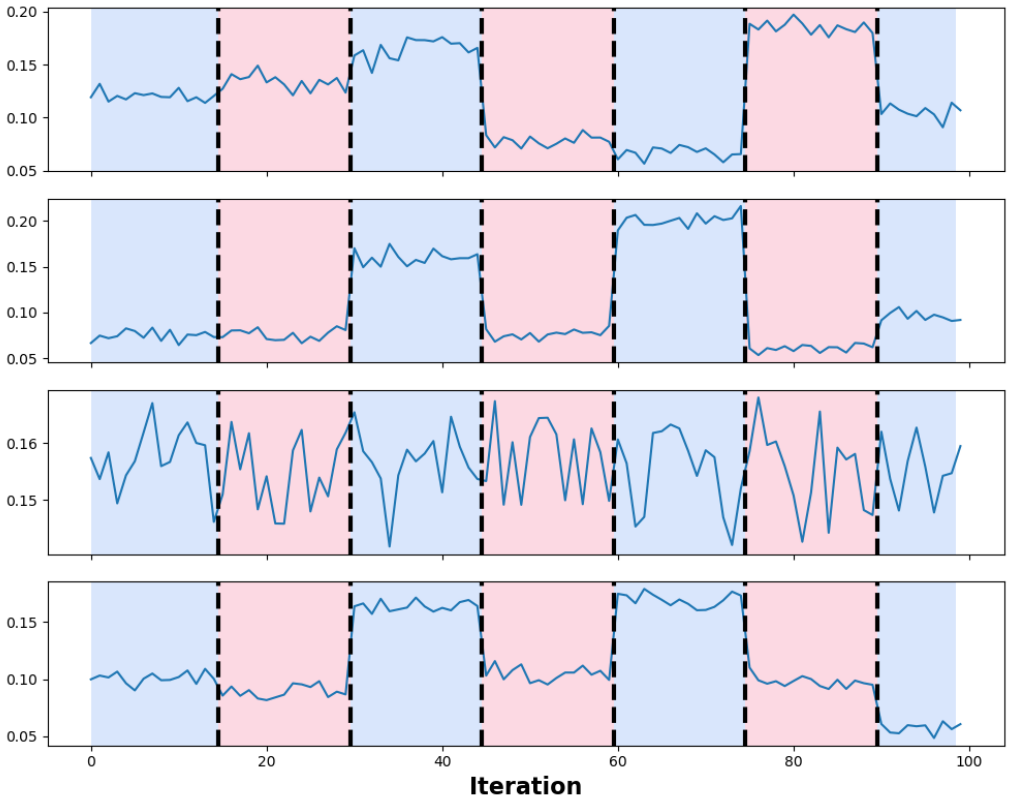
- **Μέθοδος αναζήτησης Pelt**

    1. **Συνάρτηση κόστους c_{rbf}**: Για τον συγκεκριμένο συνδυασμό και με σταδιακή αύξηση κατά 0.1 του περιορισμού (pen), προέκυψε ότι το εύρος τιμών του τελευταίου για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής είναι pen=0.2 έως 0.4. Τα τελικά αποτελέσματα απεικονίζονται στο ακόλουθο γράφημα.



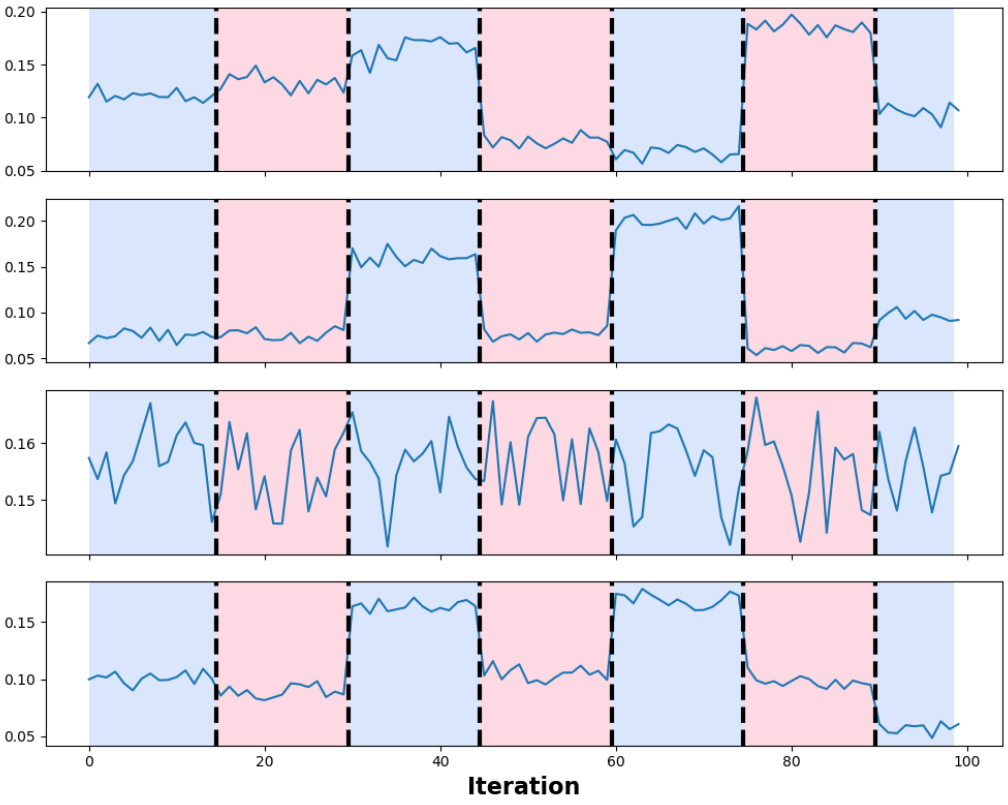
Σχήμα 6.2: Pelt, c_{rbf}, pen=0.2 έως 0.4

2. **Συνάρτηση κόστους c_{L2}**: Για τον εν λόγω συνδυασμό και με σταδιακή αύξηση κατά 0.0005 του περιορισμού (pen), παρατηρήθηκε ότι το εύρος τιμών του τελευταίου για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής είναι pen=0.001 έως 0.002. Τα τελικά αποτελέσματα παρατίθενται στο ακόλουθο γράφημα.



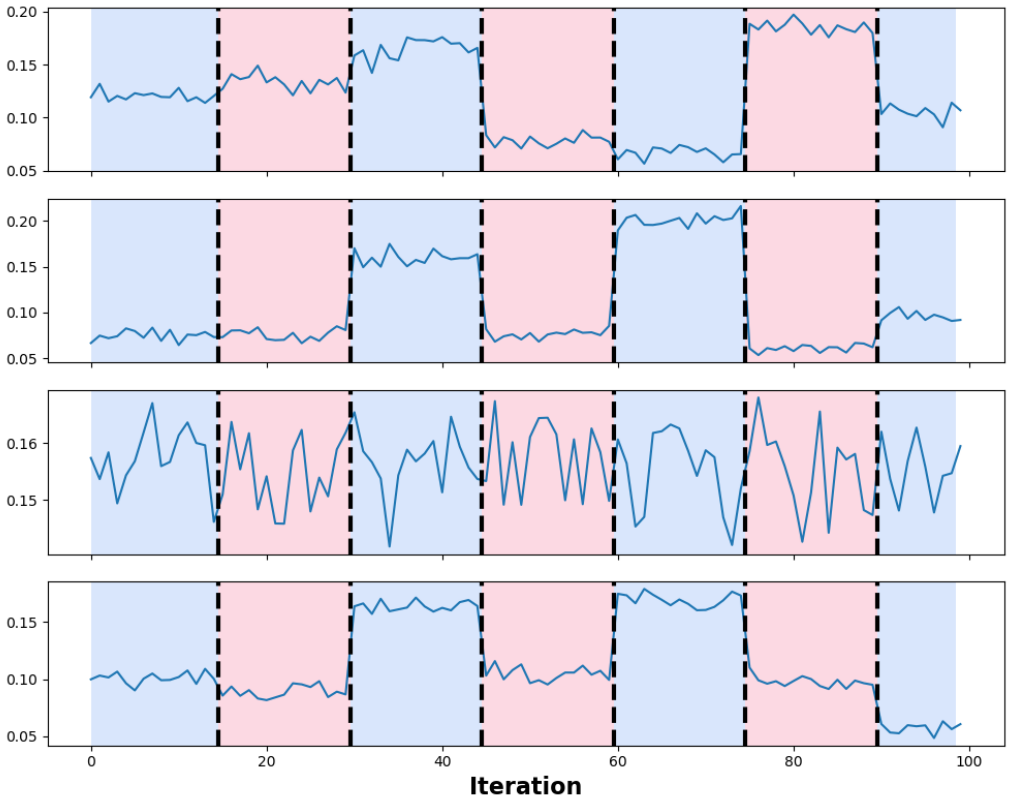
Σχήμα 6.3: Pelt, c_{L2}, pen=0.001 έως 0.002

3. **Συνάρτηση κόστους c_{rank}**: Για τον συνδυασμό αυτόν και με σταδιακή αύξηση κατά 1 του περιορισμού (pen), βρέθηκε ότι το εύρος τιμών του τελευταίου για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής είναι pen=7 έως 13. Τα τελικά αποτελέσματα παρουσιάζονται στο ακόλουθο γράφημα.



Σχήμα 6.4: Pelt, c_{rank}, pen=7 έως 13

4. **Συνάρτηση κόστους c_{L1}**: Για τον συγκεκριμένο συνδυασμό και με σταδιακή αύξηση κατά 0.01 του περιορισμού (pen), προέκυψε ότι το εύρος τιμών του τελευταίου για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής είναι pen=0.05 έως 0.18. Τα τελικά αποτελέσματα απεικονίζονται στο ακόλουθο γράφημα.



Σχήμα 6.5: Pelt, c_{L1}, pen=0.05 έως 0.18

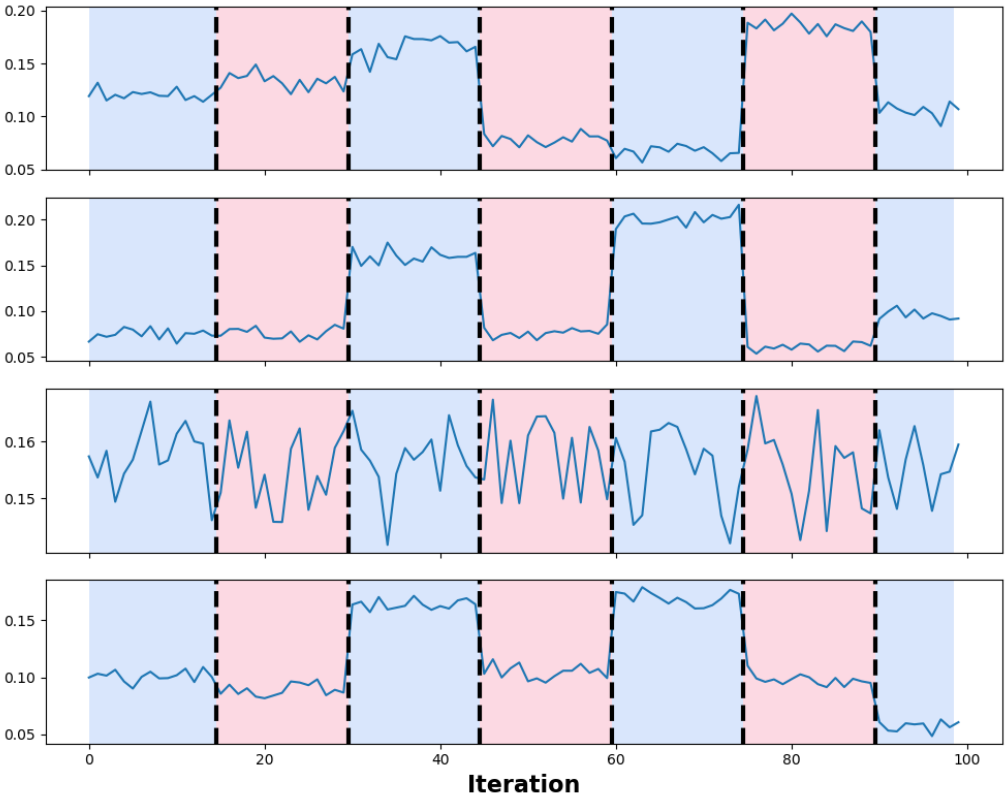
Παρατηρήσεις: Με βάση τα παραπάνω συμπεραίνει κανείς ότι, δεδομένης της μεθόδου αναζήτησης Pelt, η συνάρτηση κόστους, η οποία επιφέρει τη μεγαλύτερη ακρίβεια εκτίμησης είναι η c_{rank}. Αυτό συμβαίνει, διότι, για το ίδιο σύνολο δεδομένων και την ίδια μέθοδο αναζήτησης, η συνάρτηση αυτή επιστρέφει εξίσου ακριβή αποτελέσματα, απαιτώντας πολύ μικρότερη ¨ευαισθησία¨ κατά την εκτίμηση των σημείων αλλαγής.

- **Μέθοδος αναζήτησης Binary Segmentation (BinSeg)**

  Στην περίπτωση της μεθόδου αναζήτησης Binary segmentation (BinSeg) τα αποτελέσματα για κάθε μία από τις συναρτήσεις κόστους c_{rbf}, c_{L2}, c_{rank} και c_{L1} είναι ίδια με αυτά της μεθόδου Pelt. Επομένως, και για τη μέθοδο αναζήτησης Binary Segmentation (BinSeg), η συνάρτηση κόστους με τη μεγαλύτερη ακρίβεια εκτίμησης είναι η c_{rank}.

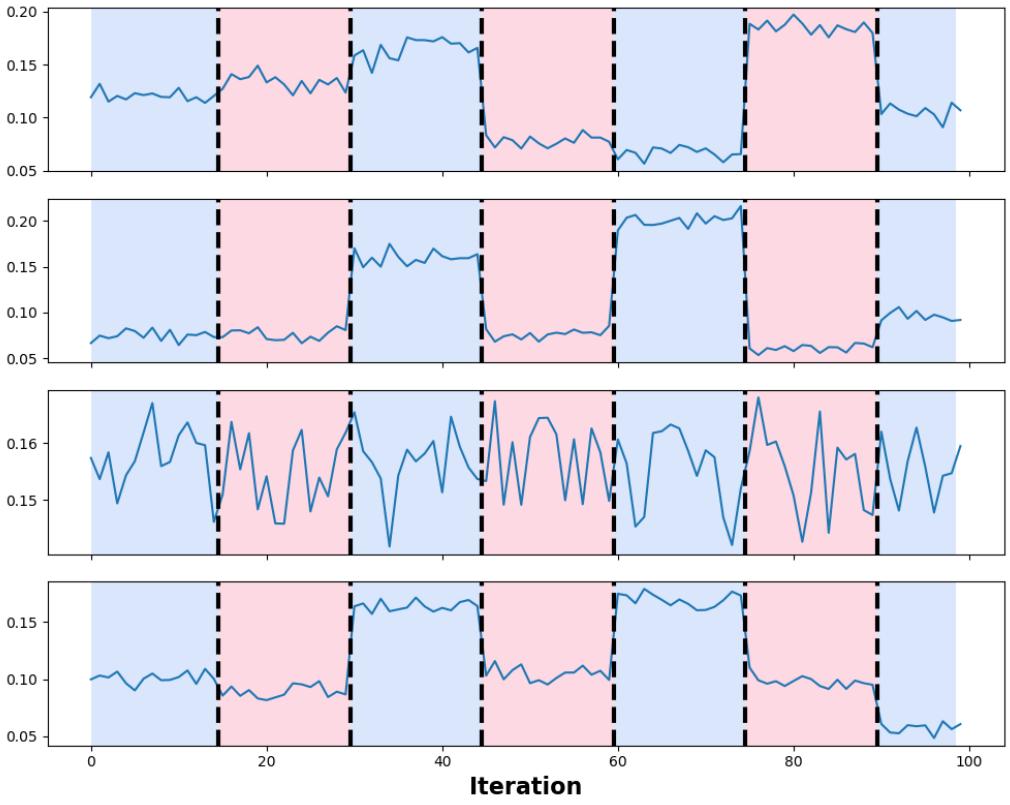
- **Μέθοδος αναζήτησης Window sliding (Win)**

  1. **Συνάρτηση κόστους c_{rbf}**: Για τον συγκεκριμένο συνδυασμό και με σταδιακή αύξηση κατά 0.1 του περιορισμού (pen), προέκυψε ότι το εύρος τιμών του τελευταίου για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής είναι pen=0 έως 0.4. Τα τελικά αποτελέσματα απεικονίζονται στο ακόλουθο γράφημα.



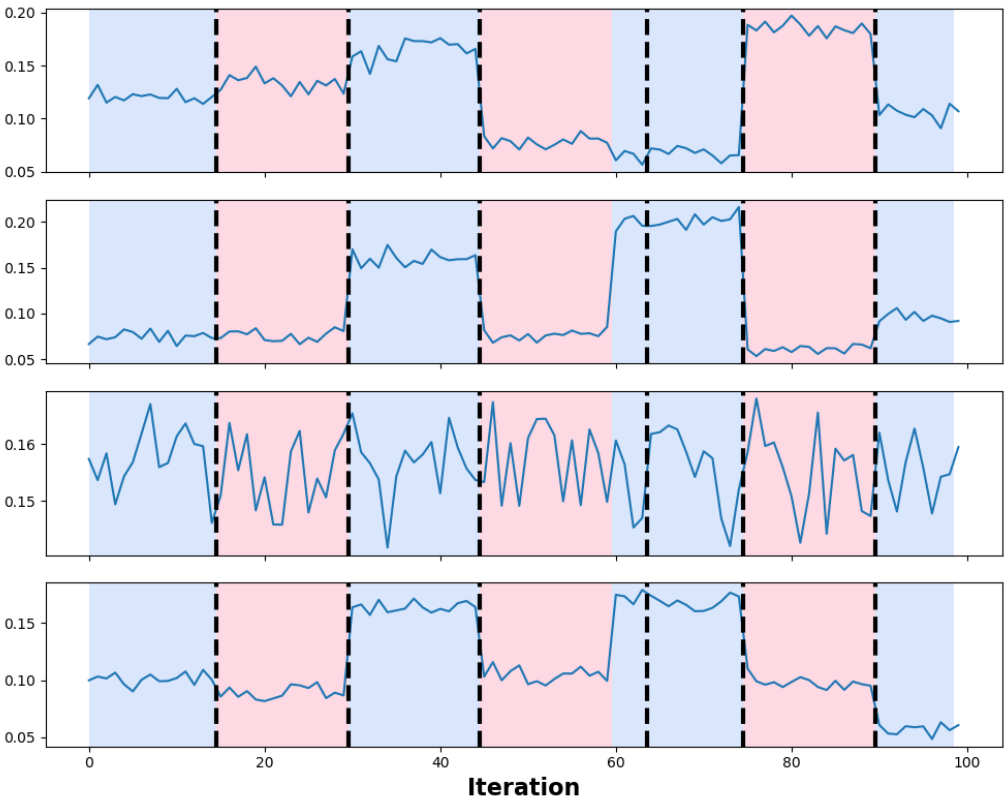
Σχήμα 6.6: Win, c_{rbf}, pen=0 έως 0.4

2. **Συνάρτηση κόστους c_{L2}**: Για τον εν λόγω συνδυασμό και με σταδιακή αύξηση κατά 0.0005 του περιορισμού (pen), παρατηρήθηκε ότι το εύρος τιμών του τελευταίου για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής είναι pen=0 έως 0.002. Τα τελικά αποτελέσματα παρατίθενται στο ακόλουθο γράφημα.

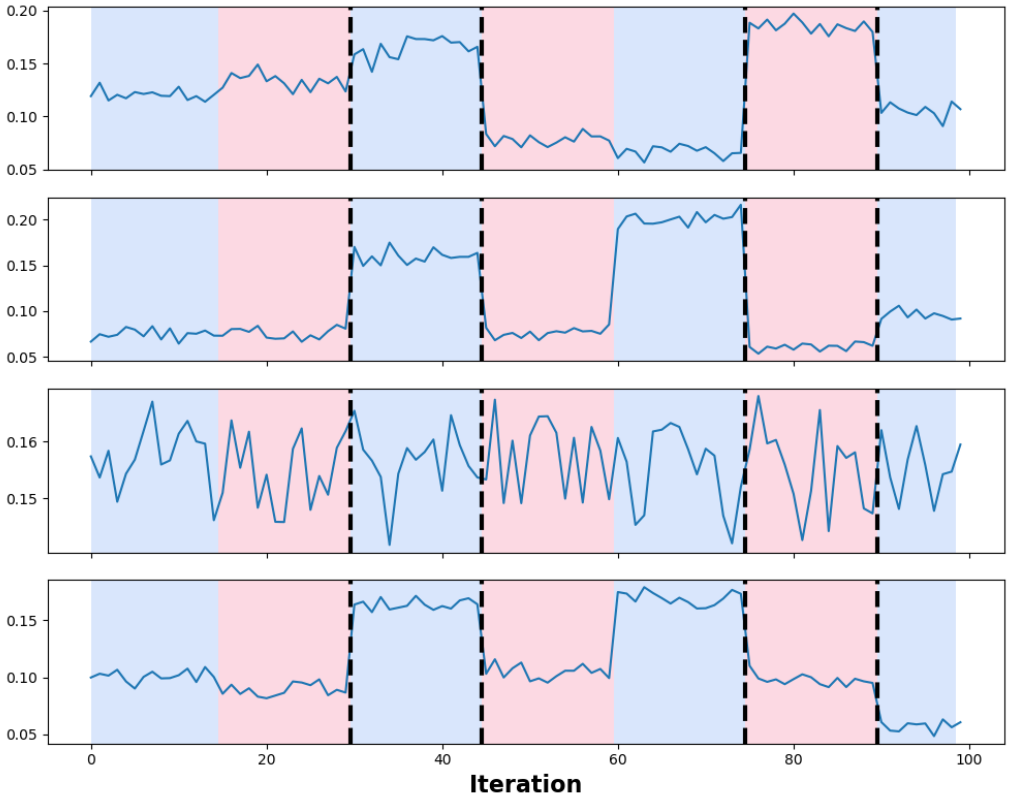


Σχήμα 6.7: Win, c_{L2}, pen=0 έως 0.002

3. **Συνάρτηση κόστους c_{rank}**: Για τον συνδυασμό αυτόν βρέθηκε ότι δεν υπάρχει εύρος τιμών για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής. Συγκεκριμένα, με σταδιακή αύξηση κατά 1 του περιορισμού (pen), παρατηρήθηκε ότι για τιμές pen=0 έως 13 εκτιμώνται με ακρίβεια όλα τα σημεία εκτός του 60, για το οποίο η αντίστοιχη εκτίμηση είναι 64. Όσο για τις τιμές του pen που είναι μεγαλύτερες του 13, η εκτίμηση είναι χειρότερη, καθώς παραλείπονται σημεία αλλαγής. Τα αποτελέσματα για pen=0 έως 13 και pen=14 παρουσιάζονται στα ακόλουθα γραφήματα.

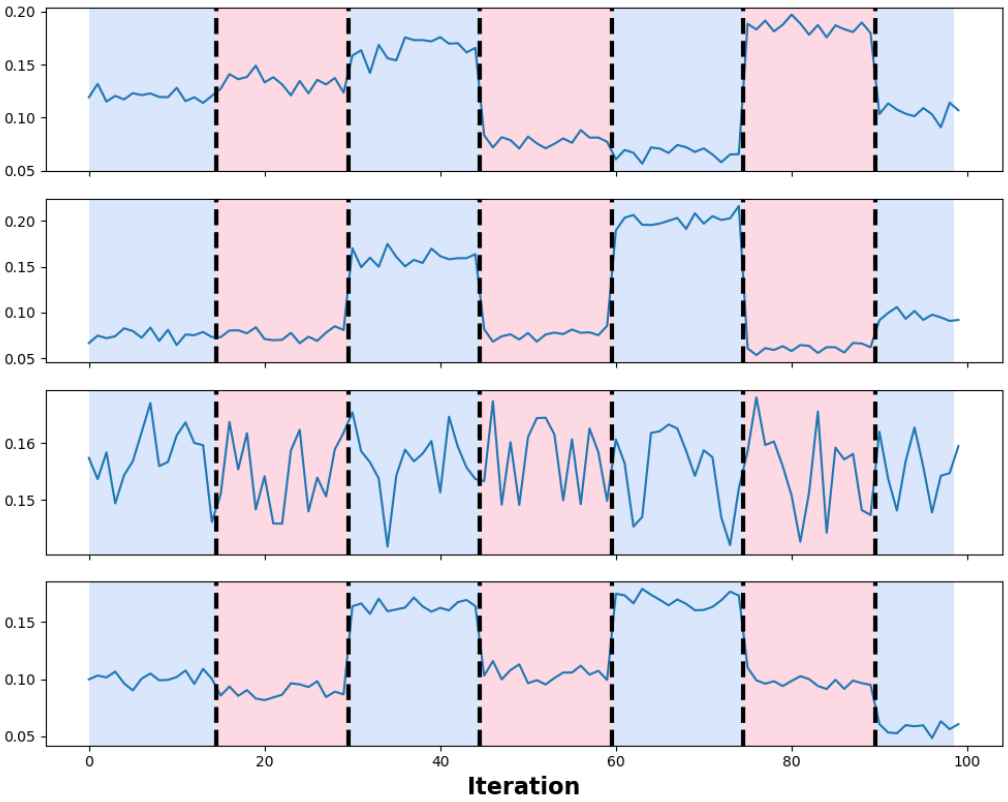


Σχήμα 6.8: Win, c_{rank}, pen=0 έως 13



Σχήμα 6.9: Win, c_{rank}, pen=14

4. **Συνάρτηση κόστους c_{L1}**: Για τον συγκεκριμένο συνδυασμό και με σταδιακή αύξηση κατά 0.01 του περιορισμού (pen), προέκυψε ότι το εύρος τιμών του τελευταίου για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής είναι pen=0 έως 0.18. Τα τελικά αποτελέσματα απεικονίζονται στο ακόλουθο γράφημα.

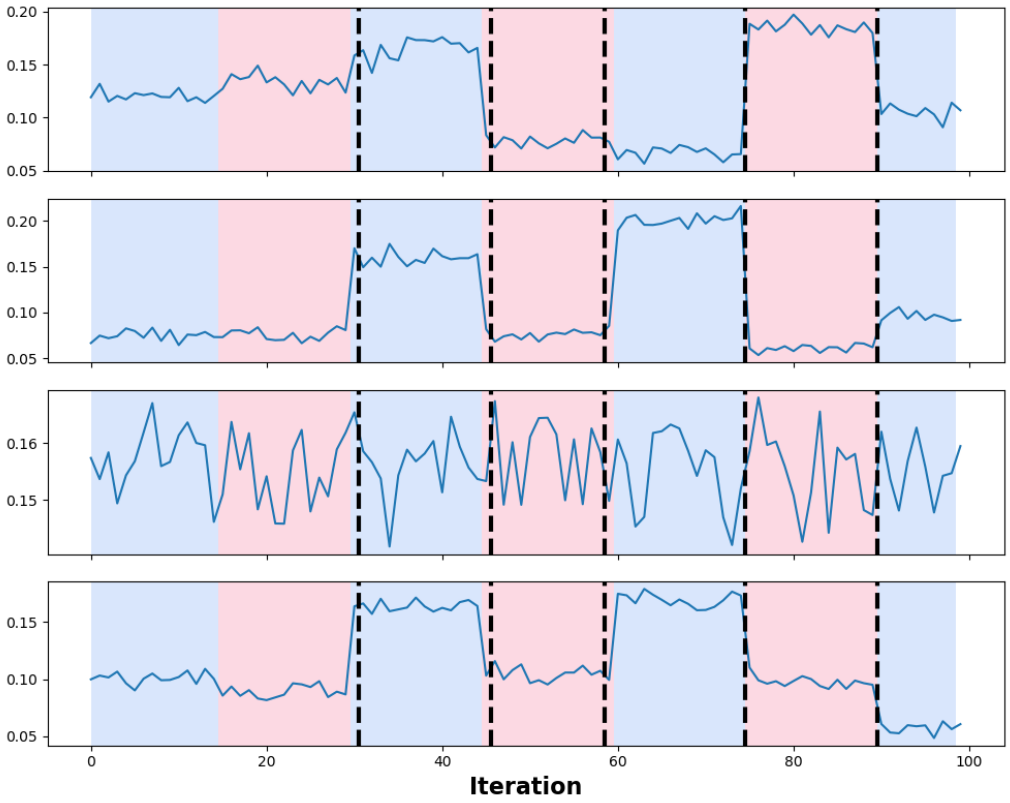


Σχήμα 6.10: Win, c_{L1}, pen=0 έως 0.18

Παρατηρήσεις: Με βάση τα παραπάνω γίνεται εμφανές ότι, δεδομένης της μεθόδου αναζήτησης Window sliding (Win), η συνάρτηση κόστους, η οποία παρέχει τη μεγαλύτερη ακρίβεια εκτίμησης είναι η c_{rbf}.

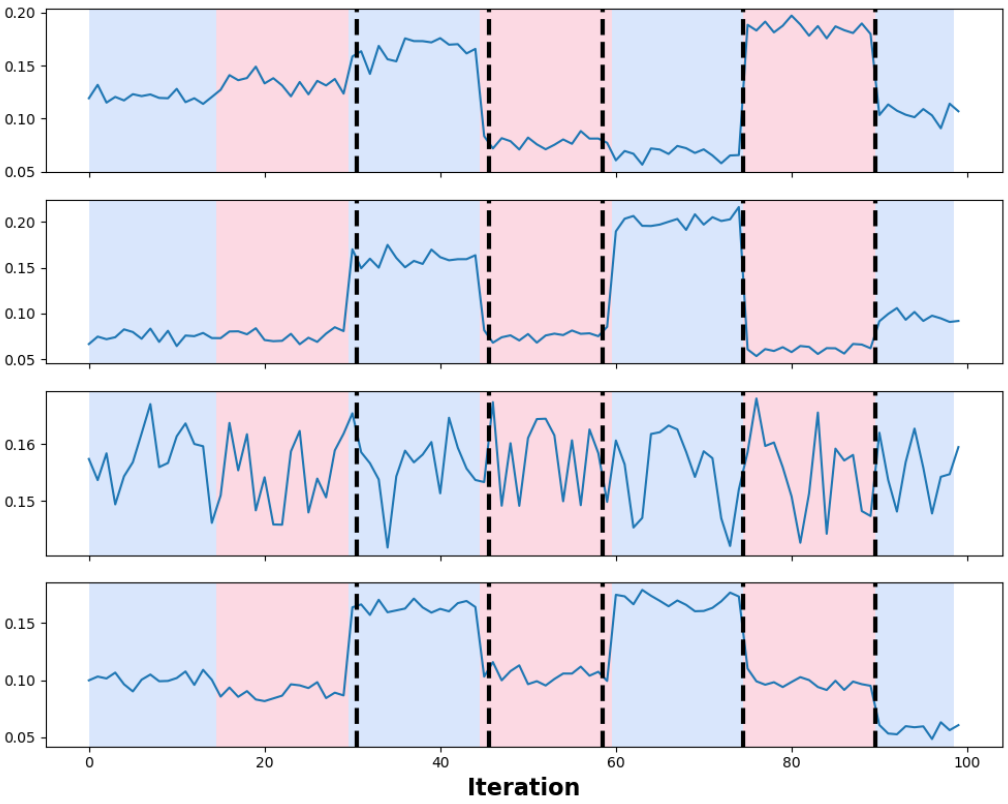
- **Μέθοδος αναζήτησης Bottom-up segmentation-BotUp**

    1. **Συνάρτηση κόστους c_{rbf}**: Για τον συνδυασμό αυτόν βρέθηκε ότι δεν υπάρχει εύρος τιμών για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής. Συγκεκριμένα, με σταδιακή αύξηση κατά 0.1 του περιορισμού (pen), παρατηρήθηκε ότι για τιμές pen=0.6 έως 6.8 τα σημεία, τα οποία εκτιμώνται είναι τα 31, 46, 59, 75, 90. Όσο για τις τιμές του pen που είναι μεγαλύτερες του 6.8, η εκτίμηση είναι χειρότερη, καθώς παραλείπονται περισσότερα σημεία αλλαγής. Τα αποτελέσματα για pen=0.6 έως 6.8 απεικονίζονται στο ακόλουθο γράφημα.



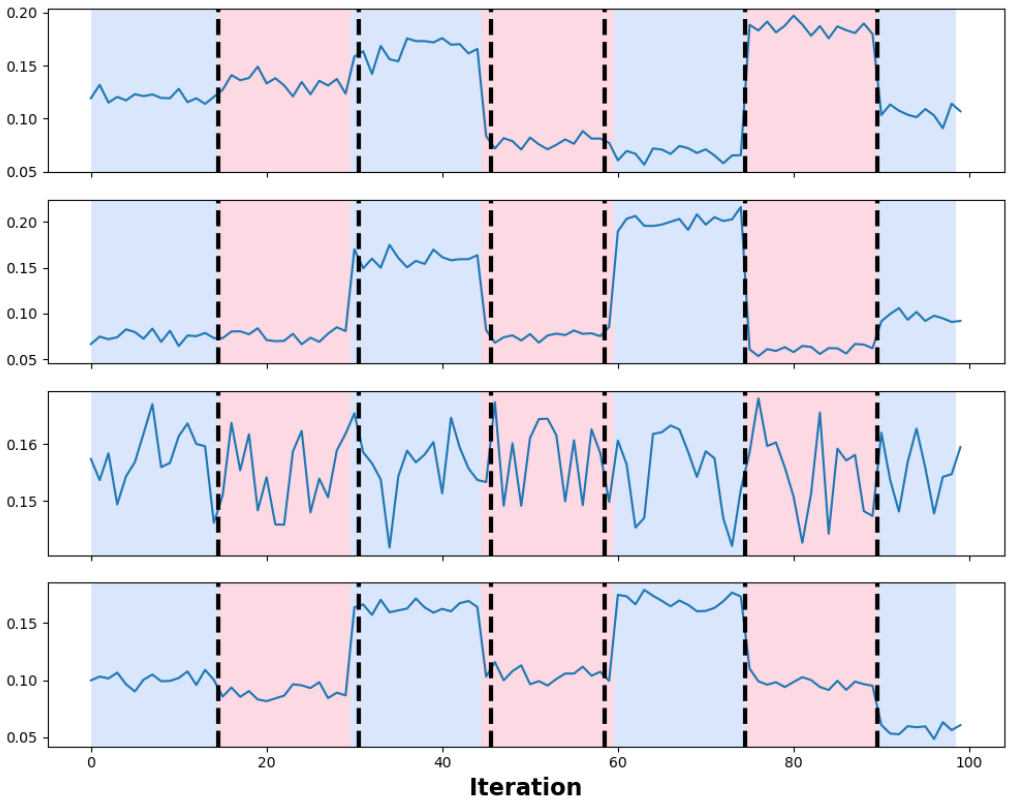
Σχήμα 6.11: BotUp, c_{rbf}, pen=0.6 έως 6.8

2. **Συνάρτηση κόστους c_{L2}**: Για τον συνδυασμό αυτόν βρέθηκε ότι δεν υπάρχει εύρος τιμών για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής. Συγκεκριμένα, με σταδιακή αύξηση κατά 0.005 του περιορισμού (pen), παρατηρήθηκε ότι για τιμές pen=0.01 έως 0.055 τα σημεία, τα οποία εκτιμώνται είναι τα 31, 46, 59, 75, 90. Όσο για τις τιμές του pen που είναι μεγαλύτερες του 0.055, η εκτίμηση είναι χειρότερη, καθώς παραλείπονται περισσότερα σημεία αλλαγής. Τα αποτελέσματα για pen=0.01 έως 0.055 παρατίθενται στο ακόλουθο γράφημα.



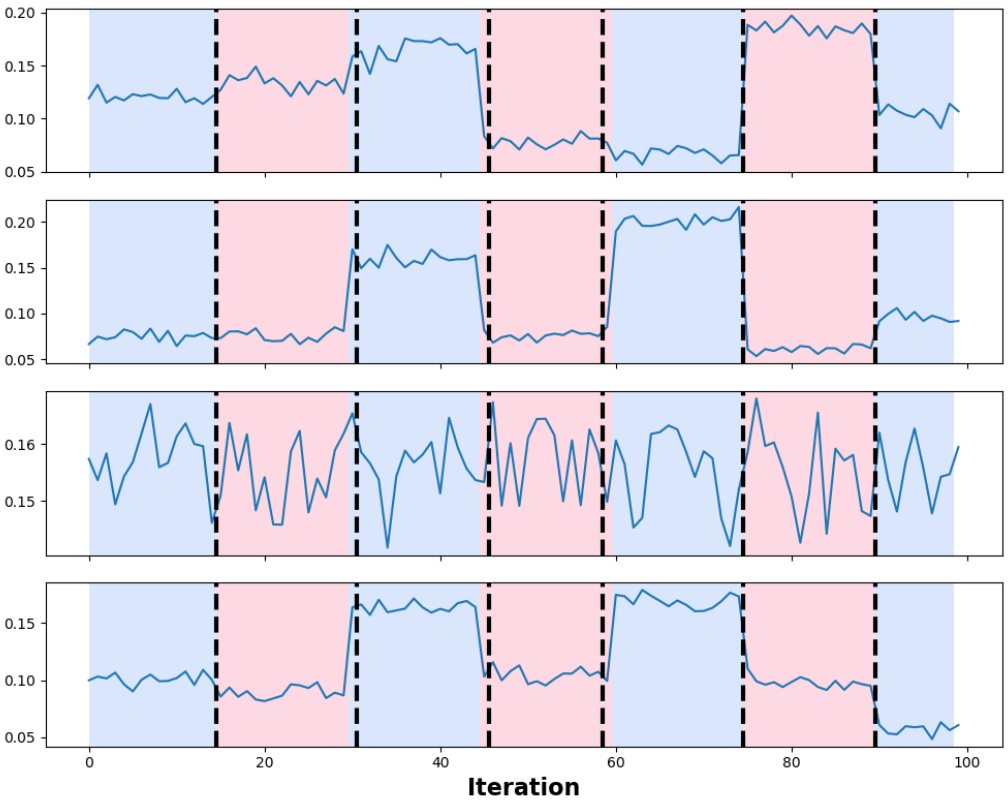
Σχήμα 6.12: BotUp, c_{L2}, pen=0.01 έως 0.055

3. **Συνάρτηση κόστους c_{rank}**: Για τον συνδυασμό αυτόν βρέθηκε ότι δεν υπάρχει εύρος τιμών για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής. Συγκεκριμένα, με σταδιακή αύξηση κατά 1 του περιορισμού (pen), παρατηρήθηκε ότι για τιμές pen=10 έως 13 τα σημεία, τα οποία εκτιμώνται είναι τα 15, 31, 46, 59, 75, 90. Όσο για τις τιμές του pen που είναι μεγαλύτερες του 13, η εκτίμηση είναι χειρότερη, καθώς παραλείπονται σημεία αλλαγής. Τα αποτελέσματα για pen=10 έως 13 παρουσιάζονται στο ακόλουθο γράφημα.



Σχήμα 6.13: BotUp, c_{rank}, pen=10 έως 13

4. **Συνάρτηση κόστους c_{L1}**: Για τον συνδυασμό αυτόν βρέθηκε ότι δεν υπάρχει εύρος τιμών για το οποίο εκτιμώνται σωστά όλα τα σημεία αλλαγής. Συγκεκριμένα, με σταδιακή αύξηση κατά 0.05 του περιορισμού (pen), παρατηρήθηκε ότι για τιμές pen=0.05 έως 0.15 τα σημεία, τα οποία εκτιμώνται είναι τα 15, 31, 46, 59, 75, 90. Όσο για τις τιμές του pen που είναι μεγαλύτερες του 0.15, η εκτίμηση είναι χειρότερη, καθώς παραλείπονται σημεία αλλαγής. Τα αποτελέσματα για pen=0.05 έως 0.15 απεικονίζονται στο ακόλουθο γράφημα.



Σχήμα 6.14: BotUp, c_{L1}, pen=0.05 έως 0.15

Παρατηρήσεις: Με βάση τα παραπάνω, γίνεται εμφανές ότι, δεδομένης της μεθόδου αναζήτησης Bottom-up segmentation (BotUp), η συνάρτηση κόστους, η οποία παρέχει τη μεγαλύτερη ακρίβεια εκτίμησης είναι η c_{rank}. Ο λόγος είναι ότι οι συναρτήσεις c_{rbf} και c_{L2} αποκλείονται, αφού αγνοούν το πρώτο σημείο αλλαγής (15), ενώ, μεταξύ των c_{rank} και c_{L1}, η c_{rank} υπερτερεί, δεδομένου ότι πετυχαίνει εξίσου ακριβή αποτελέσματα με μεγαλύτερο pen κι άρα απαιτεί πολύ μικρότερη ¨ευαισθησία¨ κατά την εκτίμηση.

Κεφάλαιο 7

Σύνοψη και Ανοιχτά Προβλήματα

Στην παρούσα διπλωματική εργασία, αρχικά, μελετήθηκε η μέθοδος της Τομογραφίας Δικτύου, καθώς επίσης κι ένας τομομογραφικός αλγόριθμος, ο οποίος έπειτα αποτέλεσε τη βάση για την πειραματική υλοποίηση μιας εφαρμογής δικτυακής επιτήρησης. Στη συνέχεια, παρουσιάστηκαν τα βασικά στοιχεία των μεθόδων στατιστικής ανάλυσης offline change point detection, καθώς ορισμένες από τις μεθόδους αυτές αξιοποιήθηκαν μετέπειτα από την εφαρμογή για την ανίχνευση σημείων αλλαγής και, κατ' επέκταση, ανωμαλιών δικτύου. Όσον αφορά το πειραματικό μέρος, πραγματοποιήθηκε μία συνοπτική επεξήγηση των εργαλείων που αξιοποιήθηκαν για την κατασκευή της δοκιμαστικής τοπολογίας, τη συλλογή μετρήσεων και τη μετέπειτα στατιστική ανάλυσή τους. Ακολούθησε η ανάλυση του αλγορίθμου της εφαρμογής και η παρουσίαση των σχετικών αποτελεσμάτων. Τέλος, πραγματοποιήθηκαν πολλαπλές εκτελέσεις με σκοπό τη συγκριτική μελέτη διαφορετικών μεθόδων offline change point detection, ως προς την ακρίβεια εκτίμησης των σημείων αλλαγής.

Συμπερασματικά, προκύπτει ότι **η παρούσα εφαρμογή δικτυακής επιτήρησης επιτυγχάνει ακριβή εκτίμηση, τόσο των εσωτερικών παραμέτρων (απώλεια πακέτων) της τοπολογίας, όσο και των εκάστοτε σημείων αλλαγής**. Συγκεκριμένα, η ακρίβεια υπολογισμού της απώλειας πακέτων, μέσω του τομογραφικού αλγορίθμου, είναι τέτοια, ώστε η μετέπειτα στατιστική ανάλυση να επιστρέφει όλα τα ζητούμενα σημεία αλλαγής, για την πλειοψηφία των μεθόδων offline change point detection. Ακόμα δε και στις περιπτώσεις, όπου ορισμένα από τα εκτιμώμενα σημεία αλλαγής δεν ταυτίζονται απόλυτα με τα πραγματικά, η απόκλιση μεταξύ τους παραμένει εξαιρετικά μικρή.

Τέτοιου είδους εφαρμογές δικτυακής επιτήρησης καταδεικνύουν τις δυνατότητες που προκύπτουν μέσα από την ανάπτυξη του πεδίου της Τομογραφίας Δικτύου. Δεδομένης, μάλιστα, της διαρκούς επέκτασης και αποκέντρωσης του Διαδικτύου, η αξία των παραπάνω δυνατοτήτων αναδεικνύεται όλο και περισσότερο. Ειδικότερα, **η έμμεση εξαγωγή της ζητούμενης πληροφορίας**, δίχως την ανάγκη συνεργασίας των εμπλεκόμενων στοιχείων του δικτύου και με υψηλή (όπως αποδείχτηκε)

ακρίβεια, **μπορεί να δώσει λύση σε ζητήματα, τα οποία καθίστανται όλο και πιο δυσεπίλυτα**. Ο έλεγχος των δικτυακών υποδομών, η ανίχνευση μεταβολών στην επίδοσή τους, καθώς και η εξασφάλιση της ακεραιότητάς τους από ενδεχόμενες επιθέσεις είναι μόνο ορισμένα από τα ζητήματα αυτά. Ζητήματα, τα οποία στο μέλλον θα είναι εξαιρετικά δύσκολο να αντιμετωπιστούν μέσω των καθιερωμένων μεθόδων επιτήρησης, καθώς οι απαραίτητες προϋποθέσεις προσβασιμότητας στο δίκτυο δεν θα υπάρχουν. Στο πλαίσιο αυτό, οι εφαρμογές Τομογραφίας Δικτύου μπορούν να παρέχουν την απαιτούμενη πληροφορία, δίχως καμία περαιτέρω προϋπόθεση, πέρα από την απλή προώθηση πακέτων.

Τέλος, όσον αφορά τις **μελλοντικές επεκτάσεις** της παρούσας διπλωματικής εργασίας, ιδιαίτερο ενδιαφέρον παρουσιάζει η δημιουργία μιας αντίστοιχης εφαρμογής δικτυακής επιτήρησης, η οποία, ωστόσο, θα εφαρμόζει **online change point detection ανάλυση**, προκειμένου να εντοπίσει τα σημεία αλλαγής στις εσωτερικές παραμέτρους της τοπολογίας. Μια τέτοια υλοποίηση θα έδινε τη δυνατότητα στον διαχειριστή του δικτύου να παρακολουθεί σε πραγματικό χρόνο τις ενδεχόμενες μεταβολές στην επίδοσή του και να λαμβάνει άμεσα τις ανάλογες αποφάσεις.

Βιβλιογραφία

[1] M. Coates, A. O. Hero III, R. Nowak, and B. Yu, "Internet tomography", *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47–65, May 2002.

[2] *What is Network Monitoring?* `https://www.cisco.com/c/en/us/solutions/automation/what-is-network-monitoring.html`, Ημερομηνία πρόσβασης: 9-8-2021.

[3] P. Qin, B. Dai, B. Huang, G. Xu, and K. Wu, "A survey on network tomography with network coding", 4, vol. 16, Institute of Electrical and Electronics Engineers (IEEE), 2014, pp. 1981–1995.

[4] G. Kakkavas, D. Gkatzioura, V. Karyotis, and S. Papavassiliou, "A review of advanced algebraic approaches enabling network tomography for future network infrastructures", *Future Internet*, vol. 12, no. 2, p. 20, Jan. 2020.

[5] M. Thottan and C. Ji, "Anomaly detection in ip networks", *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2191–2204, Aug. 2003.

[6] S. Zhao, Z. Lu, and C. Wang, "When seeing isn't believing: On feasibility and detectability of scapegoating in network tomography", in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, Jun. 2017.

[7] Δ. Γκατζιούρα, *Τομογραφία και Ουδετερότητα Δικτύου*, Οκτ. 2019.

[8] G. Kakkavas, V. Karyotis, and S. Papavassiliou, "A distance-based agglomerative clustering algorithm for multicast network tomography", in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[9] M. Coates and R. Nowak, "Network loss inference using unicast end-to-end measurement", in *ITC Seminar on IP Traffic, Measurement and Modelling, Monterey, CA*, Sep. 2000, 28:1–9.

[10] Cascares, Duffield, Horowitz, and Towsley, "Multicast-based inference of network-internal loss characteristics", *IEEE Transactions on Information Theory*, vol. 45, no. 7, pp. 2462–2480, Nov. 1999.

[11] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale IP traffic matrices from link loads", *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1, pp. 206–217, Jun. 2003.

[12] J. Cao, S. V. Wiel, B. Yu, and Z. Zhu, “A scalable method for estimating network traffic matrices from link counts”, Bell Labs, Tech. Rep., 2000.

[13] R. Castro, M. Coates, and R. Nowak, “Likelihood based hierarchical clustering”, *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2308–2321, Aug. 2004.

[14] C. Truong, L. Oudre, and N. Vayatis, “Selective review of offline change point detection methods”, *Signal Processing*, vol. 167:107299, Feb. 2020.

[15] G. A. N. Segura, A. Chorti, and C. B. Margi, “Multimetric online intrusion detection in software-defined wireless sensor networks”, in *2020 IEEE Latin-American Conference on Communications (LATINCOM)*, Nov. 2020, pp. 1–6.

[16] *Rowland, Todd. "Seminorm." From MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein*, https://mathworld.wolfram.com/Seminorm.html, Ημερομηνία πρόσβασης: 22-8-2021.

[17] *Quagga Routing Suite*, https://www.quagga.net, Ημερομηνία πρόσβασης: 13-10-2021.

[18] *Protocol Independent Multicast - PIM*, https://docs.nvidia.com/networking-ethernet-software/cumulus-linux-36/Layer-3/Protocol-Independent-Multicast-PIM/, Ημερομηνία πρόσβασης: 13-10-2021.

[19] *Quagga*, https://www.quagga.net/docs/quagga.html, Ημερομηνία πρόσβασης: 13-10-2021.

[20] *iperf-ssm*, https://github.com/noushi/iperf-ssm, Ημερομηνία πρόσβασης: 13-10-2021.

[21] *Man page of TCPDUMP*, https://www.tcpdump.org/manpages/tcpdump.1.html, Ημερομηνία πρόσβασης: 17-10-2021.

[22] *The netfilter.org "iptables" project*, https://www.netfilter.org/projects/iptables/index.html, Ημερομηνία πρόσβασης: 17-10-2021.

[23] *Controlling Network Traffic with iptables - A Tutorial*, https://www.linode.com/docs/guides/control-network-traffic-with-iptables/, Ημερομηνία πρόσβασης: 17-10-2021.

[24] *netem*, https://wiki.linuxfoundation.org/networking/netem, Ημερομηνία πρόσβασης: 22-10-2021.

[25] *Dropping Packets in Ubuntu Linux using tc and iptables*, https://sandilands.info/sgordon/dropping-packets-in-ubuntu-linux-using-tc-and-iptables, Ημερομηνία πρόσβασης: 22-10-2021.

[26] *ØMQ - The Guide*, https://zguide.zeromq.org/docs/preface/, Ημερομηνία πρόσβασης: 22-10-2021.

[27] *ØMQ - The Guide*, https://zguide.zeromq.org/docs/chapter1/, Ημερομηνία πρόσβασης: 22-10-2021.

[28] *ØMQ - The Guide*, https://zguide.zeromq.org/docs/chapter2/, Ημερομηνία πρόσβασης: 22-10-2021.

Παράρτημα Α΄

Παράρτημα

Α΄.1 Κώδικες κόμβων στο jFed

Κώδικας του κόμβου Node0

```
import zmq
import subprocess
import time
import multiprocessing
import os

TIMES=100


for i in range(TIMES):

    context = zmq.Context()

    socket0=context.socket(zmq.REP)
    socket0.connect("tcp://192.168.3.1:4444")


    #Wait for the initialization message from the controller
    print("Waiting initialisation message from the controller...")
    message=socket0.recv()
    print(f"{message}")

    done1=False

    def p1():
            subprocess.run('sudo tcpdump -nlvvv -i enp8s0f1 udp and port 5001 | tee DATA0'+str(i+1)+'.txt', shell=True)

    def p2():
            subprocess.run('iperf -c 239.1.2.3 -u -T 32 -b 2m -t 30 -i 1', shell=True)


    proc1=multiprocessing.Process(target=p1)
    proc2=multiprocessing.Process(target=p2)

    proc1.start()
    time.sleep(1)
    while(True):
        if(proc1.is_alive()):
            proc2.start()

        if(proc1.is_alive() and proc2.is_alive()):
            socket0.send(b"This is client0, I have initialized")
            break

    proc2.join()
    time.sleep(1)
```

Σχήμα Α΄.1: Κώδικας του κόμβου Node0 (μέρος α)

```
    if(not proc2.is_alive()):
        subprocess.run('sudo killall tcpdump', shell=True)
        done1=True

    if((done1==True) and (not proc2.is_alive())):
        msg=socket0.recv()
        print(f"{msg}")

        file0=open("/users/el16181n/DATA0"+str(i+1)+".txt", "r")
        data0=file0.read().rstrip('\n')
        socket0.send(("DATA0"+str(i+1)+".txt").encode("utf-8"))
        MSG=socket0.recv()
        print(f"{MSG}")
        socket0.send(data0.encode("utf-8"))
        MSG2=socket0.recv()
        print(f"{MSG2}")
        file0.close()
```

Σχήμα Α΄.2: Κώδικας του κόμβου Node0 (μέρος β)

Κώδικας του κόμβου Node1

```
import zmq
import subprocess
import time
import multiprocessing
import os

TIMES=100

for i in range(TIMES):
    context = zmq.Context()

    socket1=context.socket(zmq.REP)
    socket1.connect("tcp://192.168.2.1:5555")


    #Wait for the initialization message from the controller
    print("Waiting initialization message from the controller...")
    message=socket1.recv()
    print(f"{message}")


    def p1():
            subprocess.run('sudo tcpdump -nlvvv -i enp8s0f1 udp and port 5001 | tee DATA1'+str(i+1)+'.txt', shell=True)
    def p2():
            subprocess.run('iperf -s -P1 -u -B 239.1.2.3 -O 192.168.0.1 -X enp8s0f1 -i 1', shell=True)


    proc1=multiprocessing.Process(target=p1)
    proc2=multiprocessing.Process(target=p2)

    proc1.start()

    proc2.start()

    while(True):
        if(proc1.is_alive() and proc2.is_alive()):
            socket1.send(b"This is client1, I have initialized")
            break

    msg=socket1.recv()
    proc2.terminate()

    #subprocess.run('sudo killall tcpdump', shell=True)
    cmd='sudo killall tcpdump'
    os.system(cmd)
    print(f"{msg}")
```

Σχήμα Α΄.3: Κώδικας του κόμβου Node1 (μέρος α)

```
    file1=open("/users/el16181n/DATA1"+str(i+1)+".txt", "r")
    data1=file1.read().rstrip('\n')
    socket1.send(("DATA1"+str(i+1)+".txt").encode("utf-8"))
    MSG=socket1.recv()
    print(f"{MSG}")
    socket1.send(data1.encode("utf-8"))
    MSG2=socket1.recv()
    print(f"{MSG2}")
    file1.close()

    CMD='pkill -KILL iperf'
    os.system(CMD)
```

Σχήμα Α΄.4: Κώδικας του κόμβου Node1 (μέρος β)

Κώδικας του κόμβου Node2

```
import zmq
import subprocess
import time
import multiprocessing
import os

TIMES=100

for i in range(TIMES):
    context = zmq.Context()

    socket2=context.socket(zmq.REP)
    socket2.connect("tcp://192.168.5.1:6666")


    #Wait for the initialization message from the controller
    print("Waiting initialization message from the controller...")
    message=socket2.recv()
    print(f"{message}")


    def p1():
            subprocess.run('sudo tcpdump -nlvvv -i enp8s0f1 udp and port 5001 | tee DATA2'+str(i+1)+'.txt', shell=True)

    def p2():
        subprocess.run('iperf -s -P1 -u -B 239.1.2.3 -O 192.168.0.1 -X enp8s0f1 -i 1', shell=True)


    proc1=multiprocessing.Process(target=p1)
    proc2=multiprocessing.Process(target=p2)

    proc1.start()

    proc2.start()

    while(True):
        if(proc1.is_alive() and proc2.is_alive()):
            socket2.send(b"This is client2, I have initialized")
            break

    msg=socket2.recv()
    proc2.terminate()

    #subprocess.run('sudo killall tcpdump', shell=True)
    cmd='sudo killall tcpdump'
    os.system(cmd)
```

Σχήμα Α΄.5: Κώδικας του κόμβου Node2 (μέρος α)

```
    print(f"{msg}")

    file2=open("/users/el16181n/DATA2"+str(i+1)+".txt", "r")
    data2=file2.read().rstrip('\n')
    socket2.send(("DATA2"+str(i+1)+".txt").encode("utf-8"))
    MSG=socket2.recv()
    print(f"{MSG}")
    socket2.send(data2.encode("utf-8"))
    MSG2=socket2.recv()
    print(f"{MSG2}")
    file2.close()

    CMD='pkill -KILL iperf'
    os.system(CMD)
```

Σχήμα Α΄.6: Κώδικας του κόμβου Node2 (μέρος β)

Κώδικας του κόμβου Node3

```
import zmq
import subprocess
import time
import multiprocessing
import os

TIMES=100

for i in range(TIMES):
    context = zmq.Context()

    socket3=context.socket(zmq.REP)
    socket3.connect("tcp://192.168.9.1:7777")


    #Wait for the initialization message from the controller
    print("Waiting initialisation message from the controller...")
    message=socket3.recv()
    print(f"{message}")


    def p1():
        subprocess.run('sudo tcpdump -nlvvv -i enp8s0f0 udp and port 5001 | tee DATA3'+str(i+1)+'.txt', shell=True)

    def p2():
        subprocess.run('iperf -s -P1 -u -B 239.1.2.3 -O 192.168.0.1 -X enp8s0f0 -i 1', shell=True)


    proc1=multiprocessing.Process(target=p1)
    proc2=multiprocessing.Process(target=p2)

    proc1.start()

    proc2.start()

    while(True):
        if(proc1.is_alive() and proc2.is_alive()):
            socket3.send(b"This is client3, I have initialized")
            break

    msg=socket3.recv()
    proc2.terminate()

    #subprocess.run('sudo killall tcpdump', shell=True)
    cmd='sudo killall tcpdump'
    os.system(cmd)
```

Σχήμα Α΄.7: Κώδικας του κόμβου Node3 (μέρος α)

```
    print(f"{msg}")

    file3=open("/users/el16181n/DATA3"+str(i+1)+".txt", "r")
    data3=file3.read().rstrip('\n')
    socket3.send(("DATA3"+str(i+1)+".txt").encode("utf-8"))
    MSG=socket3.recv()
    print(f"{MSG}")
    socket3.send(data3.encode("utf-8"))
    MSG2=socket3.recv()
    print(f"{MSG2}")
    file3.close()

    CMD='pkill -KILL iperf'
    os.system(CMD)
```

Σχήμα Α΄.8: Κώδικας του κόμβου Node3 (μέρος β)

Κώδικας του κόμβου Node4

```
import zmq
import subprocess
import time
import multiprocessing
import os

TIMES=100

for i in range(TIMES):
    context = zmq.Context()

    socket4=context.socket(zmq.REP)
    socket4.connect("tcp://192.168.2.1:8888")


    #Wait for the links' loss/jitter values from the controller
    print("Waiting loss/jitter values from the controller...")

    #link 4 --- 1
    links41=socket4.recv().decode("utf-8")
    print(f"link 4 --- 1 : {links41}")
    socket4.send(b"OK with 4 --- 1")

    #link 4 --- 2
    links42=socket4.recv().decode("utf-8")
    print(f"link 4 --- 2 : {links42}")
    #socket4.send(b"OK with 4 --- 2")


    subprocess.run('sudo iptables -F', shell=True)


    subprocess.run('sudo iptables -A FORWARD -d 239.1.2.3 -o enp8s0f0 -m statistic --mode random --probability '+ links41+' -j DROP', shell=True)
    subprocess.run('sudo iptables -A FORWARD -d 239.1.2.3 -o enp8s0f1 -m statistic --mode random --probability '+ links42+' -j DROP', shell=True)

    socket4.send(b"This is client4, I have applied the loss/jitter values")
```

Σχήμα Α΄.9: Κώδικας του κόμβου Node4

Router configuration του κόμβου Node4

```
#!/bin/bash

# install quagga routing suite
sudo apt update && sudo apt install quagga -y

# enable ipv4 and ipv6 unicast forwarding
echo "net.ipv4.conf.all.forwarding=1" | sudo tee -a /etc/sysctl.conf
echo "net.ipv4.conf.default.forwarding=1" | sudo tee -a /etc/sysctl.conf
sed 's/#net.ipv6.conf.all.forwarding=1/net.ipv6.conf.all.forwarding=1/g' /etc/sysctl.conf | sudo tee /etc/sysctl.conf
echo "net.ipv6.conf.default.forwarding=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p

# enable ipv4 multicast forwarding
echo "net.ipv4.conf.all.mc_forwarding=1" | sudo tee -a /etc/sysctl.conf
echo "net.ipv4.conf.default.mc_forwarding=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p

# make configuration files
sudo touch /etc/quagga/vtysh.conf

sudo tee /etc/quagga/ospfd.conf << EOF
interface enp6s0f0
interface enp8s0f0
interface enp8s0f1
interface lo
router ospf
 passive-interface enp1s0f0
 network 192.168.4.0/24 area 0.0.0.0
 network 192.168.1.0/24 area 0.0.0.0
 network 192.168.6.0/24 area 0.0.0.0
line vty
EOF

sudo tee /etc/quagga/zebra.conf << EOF
interface enp6s0f0
 ip address 192.168.4.2/24
 multicast
interface enp8s0f1
 ip address 192.168.6.1/24
 multicast
interface enp8s0f0
 ip address 192.168.1.2/24
 multicast
interface lo
ip forwarding
ipv6 forwarding
line vty
```

Σχήμα Α΄.10: Router configuration του κόμβου Node4 (μέρος α)

```
EOF

sudo tee /etc/quagga/pimd.conf << EOF
interface enp6s0f0
 ip pim ssm
interface enp8s0f0
 ip pim ssm
 ip igmp
interface enp8s0f1
 ip pim ssm
 ip igmp
interface lo
ip multicast-routing
line vty
EOF

# change files ownwer and permissions
sudo chown quagga:quagga /etc/quagga/ospfd.conf && sudo chmod 640 /etc/quagga/ospfd.conf
sudo chown quagga:quagga /etc/quagga/pimd.conf && sudo chmod 640 /etc/quagga/pimd.conf
sudo chown quagga:quaggavty /etc/quagga/vtysh.conf && sudo chmod 660 /etc/quagga/vtysh.conf
sudo chown quagga:quagga /etc/quagga/zebra.conf && sudo chmod 640 /etc/quagga/zebra.conf

# enable and start the services
sudo systemctl --now enable zebra
sudo systemctl --now enable ospfd
sudo systemctl --now enable pimd
```

Σχήμα Α΄.11: Router configuration του κόμβου Node4 (μέρος β)

Κώδικας του κόμβου Node5

```
import zmq
import subprocess
import time
import multiprocessing
import os

TIMES=100


for i in range(TIMES):
    context = zmq.Context()

    socket5=context.socket(zmq.REP)
    socket5.connect("tcp://192.168.9.1:9999")


    #Wait for the links' loss/jitter values from the controller
    print("Waiting loss/jitter values from the controller...")

    #link 5 --- 3
    links53=socket5.recv().decode("utf-8")
    print(f"link 5 --- 3 : {links53}")
    #socket5.send(b"OK with 5 --- 3")


    subprocess.run('sudo iptables -F', shell=True)


    subprocess.run('sudo iptables -A FORWARD -d 239.1.2.3 -o enp6s0f0 -m statistic --mode random --probability '+ links53+' -j DROP', shell=True)

    socket5.send(b"This is client5, I have applied the loss/jitter values")
```

Σχήμα Α΄.12: Κώδικας του κόμβου Node5

Router configuration του κόμβου Node5

```
#!/bin/bash

# install quagga routing suite
sudo apt update && sudo apt install quagga -y

# enable ipv4 and ipv6 unicast forwarding
echo "net.ipv4.conf.all.forwarding=1" | sudo tee -a /etc/sysctl.conf
echo "net.ipv4.conf.default.forwarding=1" | sudo tee -a /etc/sysctl.conf
sed 's/#net.ipv6.conf.all.forwarding=1/net.ipv6.conf.all.forwarding=1/g' /etc/sysctl.conf | sudo tee /etc/sysctl.conf
echo "net.ipv6.conf.default.forwarding=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p

# enable ipv4 multicast forwarding
echo "net.ipv4.conf.all.mc_forwarding=1" | sudo tee -a /etc/sysctl.conf
echo "net.ipv4.conf.default.mc_forwarding=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p

# make configuration files
sudo touch /etc/quagga/vtysh.conf

sudo tee /etc/quagga/ospfd.conf << EOF
interface enp6s0f1
interface enp6s0f0
interface lo
router ospf
 passive-interface enp1s0f0
 network 192.168.8.0/24 area 0.0.0.0
 network 192.168.7.0/24 area 0.0.0.0
line vty
EOF

sudo tee /etc/quagga/zebra.conf << EOF
interface enp6s0f0
 ip address 192.168.8.1/24
 multicast
interface enp6s0f1
 ip address 192.168.7.2/24
 multicast
interface lo
ip forwarding
ipv6 forwarding
line vty
```

Σχήμα Α΄.13: Router configuration του κόμβου Node5 (μέρος α)

```
EOF

sudo tee /etc/quagga/pimd.conf << EOF
interface enp6s0f1
 ip pim ssm
interface enp6s0f0
 ip pim ssm
 ip igmp
interface lo
ip multicast-routing
line vty
EOF

# change files ownwer and permissions
sudo chown quagga:quagga /etc/quagga/ospfd.conf && sudo chmod 640 /etc/quagga/ospfd.conf
sudo chown quagga:quagga /etc/quagga/pimd.conf && sudo chmod 640 /etc/quagga/pimd.conf
sudo chown quagga:quaggavty /etc/quagga/vtysh.conf && sudo chmod 660 /etc/quagga/vtysh.conf
sudo chown quagga:quagga /etc/quagga/zebra.conf && sudo chmod 640 /etc/quagga/zebra.conf

# enable and start the services
sudo systemctl --now enable zebra
sudo systemctl --now enable ospfd
sudo systemctl --now enable pimd
```

Σχήμα Α΄.14: Router configuration του κόμβου Node5 (μέρος β)

Κώδικας του κόμβου Node6

```
import zmq
import subprocess
import time
import multiprocessing
import os

TIMES=100


for i in range(TIMES):
    context = zmq.Context()

    socket6=context.socket(zmq.REP)
    socket6.connect("tcp://192.168.3.1:1111")


    #Wait for the links' loss/jitter values from the controller
    print("Waiting loss/jitter values from the controller...")

    #link 6 --- 4
    links64=socket6.recv().decode("utf-8")
    print(f"link 6 --- 4 : {links64}")
    socket6.send(b"OK with 6 --- 4")

    #link 6 --- 5
    links65=socket6.recv().decode("utf-8")
    print(f"link 6 --- 5 : {links65}")
    #socket6.send(b"OK with 6 --- 5")


    subprocess.run('sudo iptables -F', shell=True)


    subprocess.run('sudo iptables -A FORWARD -d 239.1.2.3 -o enp6s0f0 -m statistic --mode random --probability '+ links64+' -j DROP', shell=True)
    subprocess.run('sudo iptables -A FORWARD -d 239.1.2.3 -o enp8s0f1 -m statistic --mode random --probability '+ links65+' -j DROP', shell=True)

    socket6.send(b"This is client6, I have applied the loss/jitter values")
```

Σχήμα Α΄.15: Κώδικας του κόμβου Node6

Router configuration του κόμβου Node6

```
#!/bin/bash

# install quagga routing suite
sudo apt update && sudo apt install quagga -y

# enable ipv4 and ipv6 unicast forwarding
echo "net.ipv4.conf.all.forwarding=1" | sudo tee -a /etc/sysctl.conf
echo "net.ipv4.conf.default.forwarding=1" | sudo tee -a /etc/sysctl.conf
sed 's/#net.ipv6.conf.all.forwarding=1/net.ipv6.conf.all.forwarding=1/g' /etc/sysctl.conf | sudo tee /etc/sysctl.conf
echo "net.ipv6.conf.default.forwarding=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p

# enable ipv4 multicast forwarding
echo "net.ipv4.conf.all.mc_forwarding=1" | sudo tee -a /etc/sysctl.conf
echo "net.ipv4.conf.default.mc_forwarding=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p

# make configuration files
sudo touch /etc/quagga/vtysh.conf

sudo tee /etc/quagga/ospfd.conf << EOF
interface enp6s0f1
interface enp6s0f0
interface enp8s0f1
interface lo
router ospf
 passive-interface enp1s0f0
 network 192.168.0.0/24 area 0.0.0.0
 network 192.168.4.0/24 area 0.0.0.0
 network 192.168.7.0/24 area 0.0.0.0
line vty
EOF

sudo tee /etc/quagga/zebra.conf << EOF
interface enp6s0f0
 ip address 192.168.4.1/24
 multicast
interface enp8s0f1
 ip address 192.168.7.1/24
 multicast
interface enp6s0f1
 ip address 192.168.0.2/24
 multicast
interface lo
ip forwarding
ipv6 forwarding
line vty
```

Σχήμα Α΄.16: Router configuration του κόμβου Node6 (μέρος α)

```
EOF

sudo tee /etc/quagga/pimd.conf << EOF
interface enp6s0f0
 ip pim ssm
interface enp8s0f1
 ip pim ssm
interface enp6s0f1
 ip pim ssm
 ip igmp
interface lo
ip multicast-routing
line vty
EOF

# change files ownwer and permissions
sudo chown quagga:quagga /etc/quagga/ospfd.conf && sudo chmod 640 /etc/quagga/ospfd.conf
sudo chown quagga:quagga /etc/quagga/pimd.conf && sudo chmod 640 /etc/quagga/pimd.conf
sudo chown quagga:quaggavty /etc/quagga/vtysh.conf && sudo chmod 660 /etc/quagga/vtysh.conf
sudo chown quagga:quagga /etc/quagga/zebra.conf && sudo chmod 640 /etc/quagga/zebra.conf

# enable and start the services
sudo systemctl --now enable zebra
sudo systemctl --now enable ospfd
sudo systemctl --now enable pimd
```

Σχήμα Α΄.17: Router configuration του κόμβου Node6 (μέρος β)

Κώδικας του κόμβου NodeC

```
#!/usr/bin/python3 -W ignore
import zmq
import time
import importlib
import sys

import binary_tree
import distances
import general_tree
import reciprocal_lca
import stack

import argparse
import pickle
import math

import csv
import numpy as np
from numpy.core.fromnumeric import transpose

import pandas as pd
import ruptures as rpt
import matplotlib.pyplot as plt

TIMES=100

rowlist_loss = [["Edge", "Length", "Loss Rate(%)"]]

rowlist_loss.pop(0)

parentdict=dict()
pathdict1=dict()
pathdict=dict()


# draw a sample from a beta distribution with mean mu
def draw(rng, mu, phi):
    alpha = phi * mu
    beta = phi * (1 - mu)
    return rng.beta(alpha, beta)


#create the generator
rng = np.random.default_rng()

#the iterations that are change points
changepoints=[15,30,45,60,75,90]
```

Σχήμα Α΄.18: Κώδικας του κόμβου NodeC (μέρος α)

```
#holds the mean rates of internal links
means=dict()
means['4 --- 1']=rng.uniform(0.05, 0.2)
means['4 --- 2']=rng.uniform(0.05, 0.2)
means['5 --- 3']=rng.uniform(0.05, 0.2)
means['6 --- 4']=rng.uniform(0.05, 0.2)
means['6 --- 5']=rng.uniform(0.05, 0.2)


for i in range(TIMES):

    val4 = False
    val5 = False
    val6 = False

    ack1 = False
    ack2 = False
    ack3 = False

    ready = False

    packet0 = False
    packet1 = False
    packet2 = False
    packet3 = False

    context = zmq.Context()

    socket1=context.socket(zmq.REQ)
    socket1.bind("tcp://192.168.2.1:5555")

    socket2=context.socket(zmq.REQ)
    socket2.bind("tcp://192.168.5.1:6666")

    socket3=context.socket(zmq.REQ)
    socket3.bind("tcp://192.168.9.1:7777")

    socket0=context.socket(zmq.REQ)
    socket0.bind("tcp://192.168.3.1:4444")

    socket4=context.socket(zmq.REQ)
    socket4.bind("tcp://192.168.2.1:8888")

    socket5=context.socket(zmq.REQ)
    socket5.bind("tcp://192.168.9.1:9999")
```

Σχήμα Α΄.19: Κώδικας του κόμβου NodeC (μέρος β)

```
socket6=context.socket(zmq.REQ)
socket6.bind("tcp://192.168.3.1:1111")



#check if this iteration is a change point
if i in changepoints:
    change=True
else:
    change=False


#Send the loss/jitter values to the intermediate nodes
print("Sending the loss/jitter value to client4....")
#link 4 --- 1
if change:
    means['4 --- 1']=rng.uniform(0.05, 0.2)

links41 = draw(rng, means['4 --- 1'], 10000)

links41=format(links41, ".3f")
socket4.send(links41.encode("utf-8"))
ok41=socket4.recv()

#link 4 --- 2
if change:
    means['4 --- 2']=rng.uniform(0.05, 0.2)

links42=draw(rng, means['4 --- 2'], 10000)

links42=format(links42, ".3f")
socket4.send(links42.encode("utf-8"))
ok42=socket4.recv()
val4=True


if(val4==True):
    print("Sending the loss/jitter value to client5....")
    #link 5 --- 3
    links53=draw(rng, means['5 --- 3'], 10000)

    links53=format(links53, ".3f")
    socket5.send(links53.encode("utf-8"))
    ok53=socket5.recv()
    val5=True
```

Σχήμα Α΄.20: Κώδικας του κόμβου NodeC (μέρος γ)

```
if(val5==True):
    print("Sending the loss/jitter value to client6....")
    #link 6 --- 4
    if change:
        means['6 --- 4']=rng.uniform(0.05, 0.2)

    links64=draw(rng, means['6 --- 4'], 10000)

    links64=format(links64, ".3f")
    socket6.send(links64.encode("utf-8"))
    ok64=socket6.recv()

    #link 6 --- 5
    links65=draw(rng, means['6 --- 5'], 10000)

    links65=format(links65, ".3f")
    socket6.send(links65.encode("utf-8"))
    ok65=socket6.recv()
    val6=True


#Send the initialization message to clients 0-3
if(val6==True):
    print("Sending initialization message to client1....")
    socket1.send(b"Client1, you can initialize")
    message1=socket1.recv()
    print(f"{message1}")
    ack1=True


    if(ack1==True):
        print("Sending initialization message to client2....")
        socket2.send(b"Client2, you can initialize")
        message2=socket2.recv()
        print(f"{message2}")
        ack2=True


    if(ack2==True):
        print("Sending initialization message to client3....")
        socket3.send(b"Client3, you can initialize")
        message3=socket3.recv()
        print(f"{message3}")
        ack3=True
```

Σχήμα Α΄.21: Κώδικας του κόμβου NodeC (μέρος δ)

```
while (True):
    if(ack1==True and ack2==True and ack3==True):
        print("All set, sending now to client0....")
        socket0.send(b"Client0, you can initialize")
        message0=socket0.recv()
        print(f"{message0}")
        ready=True
        break

if(ready==True):
    socket0.send(b"Client0, you can begin to send your capture file!!!")
    print("Waiting for client0 capture file...")
    filename0=socket0.recv().decode("utf-8")
    file0=open(filename0, "w")
    socket0.send(b"Got the filename, waiting for the file's data...")
    data0=socket0.recv().decode("utf-8")
    file0.write(data0)
    socket0.send(b"Got the file's data")
    file0.close()
    print("Got the capture file of client0!!!")
    packet0=True


    if(packet0==True):
        socket1.send(b"Client1, you can begin to send your capture file!!!")
        print("Waiting for client1 capture file...")
        filename1=socket1.recv().decode("utf-8")
        file1=open(filename1, "w")
        socket1.send(b"Got the filename, waiting for the file's data...")
        data1=socket1.recv().decode("utf-8")
        file1.write(data1)
        socket1.send(b"Got the file's data")
        file1.close()
        print("Got the capture file of client1!!!")
        packet1=True



    if(packet1==True):
        socket2.send(b"Client2, you can begin to send your capture file!!!")
        print("Waiting for client2 capture file...")
        filename2=socket2.recv().decode("utf-8")
        file2=open(filename2, "w")
        socket2.send(b"Got the filename, waiting for the file's data...")
        data2=socket2.recv().decode("utf-8")
        file2.write(data2)
```

Σχήμα Α΄.22: Κώδικας του κόμβου NodeC (μέρος ε)

```
            socket2.send(b"Got the file's data")
            file2.close()
            print("Got the capture file of client2!!!")
            packet2=True


        if(packet2==True):
            socket3.send(b"Client3, you can begin to send your capture file!!!")
            print("Waiting for client3 capture file...")
            filename3=socket3.recv().decode("utf-8")
            file3=open(filename3, "w")
            socket3.send(b"Got the filename, waiting for the file's data")
            data3=socket3.recv().decode("utf-8")
            file3.write(data3)
            socket3.send(b"Got the file's data")
            file3.close()
            print("Got the capture file of client3!!!")
            packet3=True


        if(packet3==True):
            #Set the arguments for the tomography code
            verbose=True
            reduction_formula='midpoint'
            src_file='DATA0'+str(i+1)+'.txt'
            dst_file= ('DATA1'+str(i+1)+'.txt', 'DATA2'+str(i+1)+'.txt', 'DATA3'+str(i+1)+'.txt')
            threshold=1e-9

            #Calculate the distances matrix
            dist_mat = distances.estimate_loss_distances(src_file, {k+1: v for k, v in enumerate(dst_file)})


            if verbose:
                print('\nMatrix of distances between terminal nodes:\n')
                print(dist_mat)

            threshold = float(threshold)
            tree = reciprocal_lca.rec_lca(dist_mat, threshold, reduction_formula, verbose)


            if verbose:
                print('\nLogical Routing Tree:')
                print('\nNodes:', [n.get_ID() for n in tree.get_nodes()])
```

Σχήμα Α΄.23: Κώδικας του κόμβου NodeC (μέρος στ)

```
    for s, r, l in sorted(tree.get_edges(), key=lambda x: (x[0].get_ID(), x[1].get_ID())):
        print('Edge: {} --- {} [Length: {:.4f}] [Loss Rate (%): {:.3%}]'.format(s.get_ID(), r.get_ID(), l, 1-10**(-1)))

print()



#Now check for the same intermidiate nodes with different IDs between iterations

#First, parse the estimations from terminal to a list
total_edges=tree.get_edges()
for s, r, l in sorted(tree.get_edges(), key=lambda x: (x[0].get_ID(), x[1].get_ID())):
    rowlist_loss.append(['{} --- {}'.format(s.get_ID(), r.get_ID()), '{:.4f}'.format(l), '{:.3%}'.format(1-10**(-1))])

    #Create a child-parent dictionary for each iteration and sort it by keys (childs)
    parentdict[r.get_ID()]=s.get_ID()
parentdict=dict(sorted(parentdict.items()))

print("The child-parent dictionary of iteration " + str(i)+ " is " +str(parentdict))


rowlist=[]
first=[]


rowlist=rowlist_loss


for t in range(len(rowlist)):
    first.append([t,rowlist[t][0].split(' --- '),False])


#If it's the first iteration keep the paths of the terminal nodes and parse the rowlist data directly to the csv file
if(i==0):
    for x in parentdict:
        if(x>=1 and x<=3):
            templist=[]
            templist.append(x)
            templist.append(parentdict[x])
            next=parentdict[x]
            while(next!=0):
                templist.append(parentdict[next])
                next=parentdict[next]
            pathdict1[x]=templist
    print(pathdict1)
    print('\n')
```

Σχήμα Α΄.24: Κώδικας του κόμβου NodeC (μέρος ζ)

```
                rowlistcsv=[]
                errorlist=[]
                correctlist=[]

                for k in range(len(rowlist)):
                    rowlist[k][2]=rowlist[k][2].replace('%','')
                    rowlistcsv.append([rowlist[k][0], rowlist[k][2]])

                rowlistcsv=np.array(rowlistcsv).T.tolist()

                #parse the list in the csv file
                with open('estimations.csv', 'w', newline='') as estimations:
                    csv_writer=csv.writer(estimations)
                    csv_writer.writerows(rowlistcsv)


                #prepare for the next iterations
                parentdict.clear()
                rowlist.clear()


            #For the next iterations correct the intermidiate nodes according to the first iteration
            elif(i!=0):
                for x in parentdict:
                    if(x>=1 and x<=3):
                        templist=[]
                        templist.append(x)
                        templist.append(parentdict[x])
                        next=parentdict[x]
                        while(next!=0):
                            templist.append(parentdict[next])
                            next=parentdict[next]
                        pathdict[x]=templist
                print(pathdict)
                print('\n')

                for x in pathdict:
                    if(pathdict[x]!=pathdict1[x]):
                        #Create the links that should be changed
                        for k in reversed(range(1,len(pathdict[x]))):
                            error1=pathdict[x][k]
                            error2=pathdict[x][k-1]
                            errorlist.append([str(error1),str(error2)])

                        #Create the correct links
                        for k in reversed(range(1,len(pathdict1[x]))):
```

Σχήμα Α΄.25: Κώδικας του κόμβου NodeC (μέρος η)

```
                correct1=pathdict1[x][k]
                correct2=pathdict1[x][k-1]
                correctlist.append([str(correct1),str(correct2)])
            print(str(errorlist)+ " must be replaced by "+ str(correctlist)+ " for terminal node " + str(x))
            print('\n')


            #Now correct the rowlist
            for m in range(len(first)):
                for n in range(len(errorlist)):
                    if(first[m][1]==errorlist[n] and first[m][2]==False):
                        first[m][1]=correctlist[n]
                        first[m][2]=True
                        break
            errorlist.clear()
            correctlist.clear()


    for i in range(len(rowlist)):
        rowlist[i][0]=first[i][1][0]+' --- '+first[i][1][1]

    rowlisttemp=[]

    for i in range(len(rowlist)):
        rowlist[i][2]=rowlist[i][2].replace('%','')
        rowlisttemp.append([rowlist[i][0], rowlist[i][2]])

    rowlisttemp=np.array(rowlisttemp).T.tolist()
    rowlisttemp.pop(0)
    rowlisttemp=[item for sublist in rowlisttemp for item in sublist]

    rowlistcsv.append(rowlisttemp)

    #parse the list in the csv file
    with open('estimations.csv', 'w', newline='') as estimations:
        csv_writer=csv.writer(estimations)
        csv_writer.writerows(rowlistcsv)


    #prepare for the next iterations
    parentdict.clear()
    rowlist.clear()
    pathdict.clear()
```

Σχήμα Α΄.26: Κώδικας του κόμβου NodeC (μέρος 9)

```
#parse the final csv file to a DataFrame
df = pd.read_csv('estimations.csv',dtype = float, float_precision='high')

for i in range(TIMES):
    df.loc[i]=df.loc[i].apply(lambda x: x /100)

#delete the zero column that refers to the link connecting the source
df = df.loc[:, (df != 0).any(axis=0)]

print(df)
print('\n')

#Create the numpy array that will be used as input for the ruptures
signal=df.to_numpy(dtype=float)
#print(signal)

#Change point detection
model="rbf"
algo = rpt.Pelt(model=model, jump=1, min_size=1).fit(signal)
my_bkps = algo.predict(pen=0.3)

#Show results
bkps=[15,30,45,60,75,90,99]
rpt.show.display(signal, bkps, my_bkps)
plt.xlabel ("Iteration", fontsize=16, fontweight='bold')
plt.tight_layout()
print("Detected change points at iterations:", my_bkps[:-1])

plt.show()
```

Σχήμα Α΄.27: Κώδικας του κόμβου NodeC (μέρος ι)